Supply Management for Cost Minimization in Assembly Systems with Random Component Yield Times

Jean-Marie Proth, Gilles Mauroy, Yorai Wardi, Chengbin Chu, Xiaolan Xie

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Supply Management for Cost Minimization in Assembly Systems with Random Component Yield Times

Jean-Marie Proth - Gilles Mauroy - Yorai Wardi
Chengbin Chu - Xiaolan Xie

N° 3104
Février 1997

THÈME 4

OPTIMISATION DE LA GESTION DES APPROVISIONNEMENTS DANS DES SYSTEMES D'ASSEMBLAGE AVEC TEMPS DE REAPPROVISIONNEMENT ALEATOIRES

Jean-Marie Proth, Gilles Mauroy[1], Yorai Wardi[2], Chengbin Chu et Xiaolan Xie[3 4]

Résumé. Ce papier concerne la gestion des stocks dans des systèmes d'assemblage. Il s'agit de minimiser le coût de production composé d'un coût de stockage et d'un coût de rupture. Ces systèmes fabriquent de nombreux produits assemblés à partir de différents composants, et ils fonctionnent de manière cyclique. En début de cycle, deux décisions sont prises : on définit la quantité de produits à assembler durant le cycle suivant et le nombre de composants de chaque type à commander. Le coût de rupture concerne les produits finis et le coût de stockage concerne les composants.

La difficulté du problème réside dans les aléas qui apparaissent sur la demande de produits finis et sur les délais de livraison. Ces aléas peuvent être modélisés de manière probabiliste et, par conséquent, le problème à résoudre est un problème stochastique, mais ce problème est extrêmement difficile du fait de la non linéarité des équations obtenues, de la présence de variables discrètes et continues, et du grand nombre de ces variables, même pour des problèmes de taille moyenne.

Notre approche consiste à définir d'abord quelques paramètres de contrôle, ce qui réduit le nombre de variables du problème d'optimisation, et ensuite à résoudre le problème d'optimisation en utilisant les techniques d'optimisation sophistiquées avec une modélisation heuristique. Nous illustrerons, à l'aide d'exemples numériques, la manière dont ce difficile problème est résolu. On pourra observer que les résultats obtenus sont satisfaisants.

Mots-clefs. Gestion des réapprovisionnements, Optimisation stochastique, Systèmes à événements discrets, Modèle continu, Gradient.

---

[1] Les activités de recherche de cet auteur sont financées par le "Julian Hightower Chair" du "Georgia Institute of Technology".

[2] Auteur à contacter.

[3] Les premier, quatrième et cinquième auteurs appartiennent à l'INRIA-Lorraine, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, France. Les deuxième et troisième auteurs appartiennent à Georgia Institute of Technology, School of Electrical Engineering, Atlanta, GA 30332-0230, USA.

[4] Ce travail a été financé dans le cadre de l'accord NSF (National Science Foundation) - INRIA.

SUPPLY MANAGEMENT FOR COST MINIMIZATION IN ASSEMBLY SYSTEMS WITH RANDOM COMPONENT YIELD TIMES

J.M. Proth, G. Mauroy,[1] Y. Wardi,[2], C. Chu, and X.L. Xie[3]

Abstract. This paper is concerned with inventory control in assembly systems for minimizing production costs. The system manufactures multiple products assembled from various components, and it operates according to a cyclic schedule. At the start of each cycle time, two decisions are made: the product volumes to be assembled during the current cycle, and the component-stock levels to be ordered. For a given decision, there is an associated cost incurred by backlogging of the finished products on one hand, and the component inventory holding cost, on the other hand. The objective here is to balance the two costs so as to minimize their sum.

One complicating factor stems from uncertainties in both product demand levels and components yield times. These uncertainties can be modeled by probabilistic means, and hence the cost minimization problem becomes a stochastic problem. This problem can be quite difficult due to the nonlinearity of the equations involved, the mix of integer and continuous parameters, and their large number in moderate-size problems.

Our approach in this paper is to first define certain control parameters and thus reduce the number of the variables involved in the optimization problem, and then solve the latter problem by using sophisticated optimization techniques in conjunction with heuristic modeling. We will demonstrate, by numerical means, the resolution of fairly difficult problems and thus establish the viability of the proposed numerical techniques.

Key words. Supply management, stochastic optimization, discrete event systems, continuous flow models, gradient descent.

---

[1]Research of this author has been supported by the Julian Hightower Chair at the Georgia Institute of Technology.

[2]Corresponding author.

[3]The first, fourth abd fifth authors are with INRIA-Lorraine, Technopole Metz 2000, 4 rue Marconi, 57070 Metz, France. The second and third authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, U.S.A.

1 Introduction

With the increasing globalization of the manufacturing competitive arena and the resulting shifts from economies of scale to economies of scope, companies must carefully plan their production schedules in order to reduce costs while satisfying customers' demands on time. This may necessitate closer consideration of various cost measures associated with the production operation, involving supply and material management policies, inventory control, and other factors concerning material flow. One aspect of such cost containment involves the timing associated with the movement of parts and products, as this impacts inventory-holding as well as product-backlogging costs.

Such design and control problems often can be cast into the form of optimization problems, where one seeks a "best" parameter from a given set in order to minimize, or contain the resulting cost. These optimization problems typically pose a number of difficulties due to the presence of large numbers of continuous and discrete variables, the presence of uncertain or random elements, and the lack of closed-form expressions for the cost measures considered. Consequently, effective solution techniques may have to involve careful modeling, presentation of the problem in a way where computer algorithms can be coded and run with relative ease, and a slew of sophisticated, but sometime ad-hoc based mathematical programming algorithms.

This paper concerns the problem of supply management and inventory/backlogging control in assembly systems with random components yield times. We consider a cyclic manufacturing system where, at each production cycle, various products have to be assembled from a list of components. At the start of a cycle, two decisions have to be made:

1. How many products of a given type to assemble.
2. How many components to order.

Parts (components) ordered at the start of a given cycle can arrive during that cycle or at later times, and in either case, they cannot be used for the assembly during that cycle. In other words, the production has to be based on the stocks available at the start of a cycle. Ordering too many components results in excessive inventory buildup, and too few, in

product backlogging. The objective here is to find a balance between the inventory holding costs and the backlogging costs.

Inventory holding costs generally can be quantified by fairly simple equations, say it costs a_j dollars to hold one type-j part during one production cycle, and the total cost is linear. Regarding product backlogging costs, quantification can be more difficult if possible at all. There are qualitative effects like customers' dissatisfaction and loss, certain contract items pertaining to on-time delivery, etc.. In other situations, however, it may be possible to assign a simple, linear quantity to the backlogging costs. For instance, when a product available late is simply not sold, the resulting loss of revenue can be counted as a cost item. In the forthcoming discussion we will assume such a linear cost measure where, for one type-i product, the backlogging cost is b_i dollars.

The supply management problem will be formulated in terms of minimizing the sum of the parts' inventory holding costs and the products' backlogging costs. The modeling and solution techniques are also applicable, *mutatis mutandis* via penalty function methods, to constrained optimization problems, where the cost is related to inventories and the constraints are on the backlogging. This can be suitable in situations where quantitative measures of the backlogging costs are unavailable, and constraints on the products' tardiness are imposed.

Our solution technique for the above-mentioned optimization problem involves modeling, parametrization of the design problem in a form amenable to computer algorithms, heuristics, and an adaptation of well-established, powerful nonlinear-programming algorithms to the stochastic environment under study. Although the problem is quite difficult to solve, the results of our numerical experiments testify to the viability of the approach proposed in this paper.

The design optimization problem considered here was formulated in [2, 3], and algorithmic approaches were presented in [3] and [10]. These algorithms were based on sample-path gradient descent, and the Infinitesimal Perturbation Analysis (IPA) technique for gradient estimation (see [7] for a discussion on IPA). [3] presented a stochastic approximation algorithm, and [10] applied alternative modeling and algorithmic methods with large stepsizes.[4]

[4]This is in contrast to the stochastic approximation approach, where the stepsizes decline to 0 at an a

This paper in essence is an extended (journal) version of the latter reference, with deeper discussion on the general methodology and various implementation aspects.

Related works on supply management have dealt mainly with supplier selection [4, 5, 6] and, in the case of random yield times, an analysis under the assumption of no conflict in the use of components, was carried out in [2]. The general problem, without any restrictions on the component yield time distributions, has been considered in [3, 10] as earlier pointed out.

The paper is organized along the following lines. Section 2 describes the problem and the main approach to it taken in [3], which forms the basis for the developments here. Section 3, following [10], presents our alternative modeling paradigm and algorithmic approach. Section 4 contains results of numerical experiments, and Section 5 concludes the paper.

2 Problem formulation

Consider a cyclic assembly manufacturing system for multiple products sharing a common component bank. Suppose there are n product types and m component types, and that the assembly of one type-i product requires $\ell_{i,j}$ type-j components, for all $j = 1, \ldots, m$. Thus we can define the assembly matrix L,

$$L = \begin{pmatrix} \ell_{1,1} & \cdot & \cdot & \cdot & \ell_{1,m} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \ell_{n,1} & \cdot & \cdot & \cdot & \ell_{n,m} \end{pmatrix}$$

where, the assembly of one type-1 product requires $\ell_{1,1}$ type-1 components, $\ell_{1,2}$ type-2 components, etc. etc..

At the start of the *kth* production cycle two decisions have to be made: (i) how many type-i products to assemble during the cycle, and (ii) how many components of the various types to order. It is assumed of course that the products' demands for a cycle are known at

priori rate. For a comprehensive treatment of the latter optimization technique, please see [9]; its aspects related to the present problem have been discussed in [3].

its start, and delivery is expected at its end. The first decision can be based on the various components' inventory levels at the start of the cycle. Regarding the second decision, one has to project into the future since components delivered during a cycle cannot be used for assembly at that cycle, but only during later production cycles. Such components, ordered at the start of the kth cycle, may arrive during that cycle and thus be available for assembly during the $(k+1)th$ cycle, but they also may arrive during the $(k+q)th$ cycle, $q = 1, 2, \ldots$, and be available during the following one.

The above decisions clearly would affect the tradeoff between the accumulation of components' inventory and products' backlogging. Let $b_i > 0$ be the per-cycle cost of one type-i product backlogging, and let $a_j > 0$ be the per-cycle cost of one type-j component inventory holding. Let v_i^k be the backlogging of type-i products at the start of the kth production cycle; this quantity does not take into account the production decision at that cycle, but it certainly accounts for the decisions made at the previous, $(k-1)th$ cycle. Let s_j^k denote the inventory of type-j components at the start of the kth cycle, after the components needed for assembly at that cycle are taken out. Thus, the total backlogging plus inventory cost associated with the kth cycle is $\sum_{i=1}^{n} b_i v_i^k + \sum_{j=1}^{m} a_j s_j^k$. Let us fix the number of production cycles under consideration for the design optimization problem at some given value of K. Then, the cost associated with the assembly during that period, denoted by G_K, is equal to

$$G_K \;=\; K^{-1} \sum_{k=1}^{K} [\sum_{i=1}^{n} b_i v_i^k + \sum_{j=1}^{m} a_j s_j^k]. \tag{2.1}$$

This cost measure is a function of the decision variables earlier mentioned, i.e., the production-volumes schedule and the components' ordering policy at each cycle. The effect of these variables on the cost can be seen via the dynamic (state) equations, next described. Let y_i^k be the number of type-i products assembled during the kth cycle, and let x_j^k denote the number of type-j components ordered at the start of the kth cycle. Due to the random component-yield times, we denote by z_j^k the number of type-j components arriving during the kth cycle. This quantity depends on the decision variables x_j^{k-q}, $q = 0, 1, 2, \ldots$, according to some probabilistic law. Finally, let d_i^k denote the demand for type-i products at the kth cycle. The dynamic equations for the inventory and backlogging levels now become

$$s_j^k \;=\; s_j^{k-1} + z_j^{k-1} - \sum_{i=1}^{n} y_i^k \ell_{i,j}, \tag{2.2}$$

$$v_i^k \;=\; v_i^{k-1} + d_i^{k-1} - y_i^{k-1}. \tag{2.3}$$

The cost G_K is a random function due to the fact that the products' demands and components' yield times are random. Let $(\Omega, \mathcal{F}, \mathbb{P})$ denote the probability space underlying realizations of G_K, and let $\mathbb{E}[\cdot]$ denote expectation in that space. The performance measure of interest is the expected-value cost, namely,

$$g_K \;:=\; \mathbb{E}[G_K],$$

and the design-optimization problem is to minimize g_K over all possible parameters $\{y_i^k\}$ and $\{x_j^k\}$.

As an optimization problem, it has a number of characteristics that make it quite hard to solve by standard mathematical-programming techniques. First, it is stochastic, and due to the lack of closed-form solution to the performance measure, may require Monte Carlo simulation for function evaluations. Moreover, it may involve too many variables having integer values: $n+m$ variables for each cycle, and hence $(n+m)K$ variables over K cycles. To ameliorate the latter difficulty, we follow the approach taken in [3], consisting of a heuristic argument for determining y_i^k, and a parametrization of the problem by a reduced number of control variables regarding the decision parameters x_j^k.

The heuristic reasoning goes as follows. An assembly of one extra type-i product would, under some circumstances, reduce the backlogging cost by b_i, and the inventory holding cost, by $\sum_{j=1}^m \ell_{i,j} a_j$. We denote the sum of these costs by ϕ_i, i.e.,

$$\phi_i \;=\; b_i + \sum_{j=1}^{m} \ell_{i,j} a_j.$$

Let us order the sequence ϕ_i, $i = 1, \ldots, n$, in nonincreasing order, and suppose that

$$\phi_1 \geq \phi_2 \ldots \geq \phi_n.$$

Now our production-volume policy states that, we assemble as many type-1 products as possible, then as many type-2 products as possible, etc.. This decision has to be made at the start of each cycle. Consider the *kth* cycle. At its start, the number of type-j components available is $s_j^{k-1} + z_j^{k-1}$, $j = 1, \ldots, m$,[5] the demands for the current cycle are d_i^k, and the

[5]The inventory level s_j^k excludes the components taken out for assembly during the cycle, and z_j^{k-1} is the number of such components that arrived during the previous cycle. Therefore, and by Eq. (2.2), $s_j^{k-1} + z_j^{k-1}$ is the number of available type-j components at the start of the *kth* cycle.

backlogging inherited from previous cycles is v_i^k. All of these quantities are known at the start of the kth cycle. To assemble as many type-1 products as possible, two constraints must be taken into account: the demand plus backlogging, and the volumes of available components. Recall that, the manufacturing of one type-1 product requires $\ell_{i,j}$ type-j components for all $j = 1, \ldots, m$. Thus, we get

$$y_1^k \;=\; min\{d_1^k + v_1^k, [\frac{s_j^{k-1} + z_j^{k-1}}{\ell_{1,j}}] \;:\; j = 1, \ldots, m\}; \tag{2.4}$$

$[\xi]$ denotes the integer part of ξ. With this value of y_1^k, we take out $\ell_{1,j} y_1^k$ type-j components for the assembly of type-1 products, and hence the remaining number of components for type-2 products is $s_j^{k-1} + z_j^{k-1} - \ell_{1,j} y_1^k$. Following a similar policy for type-2 products, etc., we get, for all $i = 2, \ldots, m$,

$$y_i^k \;=\; min\{v_i^k + d_i^k, [\frac{s_j^{k-1} + z_j^{k-1} - \sum_{v=1}^{i-1} \ell_{v,j} y_v^k}{\ell_{i,j}}] \;:\; j = 1, \ldots, m\}. \tag{2.5}$$

We point out that the above policy concerning the ordering $\phi_1 \geq \phi_2 \geq \ldots \geq \phi_n$ can be determined a priori because ϕ_i depends only on the problem's data, and hence can be computed in advance of running the algorithm.

Concerning the variables x_j^k, we parametrize the optimization problem by m hedging points, one for each component type. That is, the hedging point h_j is the inventory target level of type-j components. Such hedging points have been shown in [8] to yield optimal control of inventory levels; see [1] and the references therein for further results. Recall that the inventory level at the start of the kth cycle is s_j^k after all of the components required for the assembly at that cycle have been removed. The basic control policy is this: with h_j being the target level, if $h_j \leq s_j^k$, then order no type-j components, i.e., set $x_j^k = 0$, and if $h_j > s_j^k$, set $x_j^k = h_j - s_j^k$. In other words, we have that $x_j^k = (h_j - s_j^k)_+$, where $(\zeta)_+ := max\{\zeta, 0\}$. This policy, however, does not account for parts ordered in the past that may arrive during this cycle, and hence may result in excessive inventory levels.

To address this issue, Chu et al [3] introduced a correction parameter based on a discount factor $\alpha_j \in [0, 1]$. This factor subtracts from the target level h_j not only the inventory level s_j^k, but also a term related to the number of components, previously ordered, and expected to arrive during the current cycle. Commonly used in the field of finance, the discount factor assigns a greater value to components more recently ordered. With $U_{j,q}^k$ denoting the number

of type-j components ordered q cycles ago and expected to arrive during the present (kth) cycle, we set [3]

$$x_j^k = [h_j - (s_j^k + \sum_{q=1}^{\infty}(\alpha_j)^q U_{j,q}^k)]_+, \tag{2.6}$$

where $[\xi]$ denotes the integer part of ξ.

A somewhat simpler way to account for the components that have been ordered and not yet arrived is to consider the number of such late parts, denoted by u_j^k, instead of their expected values. In this case, we set [10]

$$x_j^k = [h_j - s_j^k - \alpha_j u_j^k]_+. \tag{2.7}$$

This, in contrast to the term in (2.6), assigns an equal weight to latent components ordered in the recent past or much earlier. We can modify (2.7) via

$$x_j^k = [h_j - s_j^k - \sum_{q=1}^{\infty}(\alpha_j)^q u_{j,q}^k]_+, \tag{2.8}$$

where $u_{j,q}^k$ is the number of type-j components ordered q cycles ago and not yet arrived.

The purpose of the terms involving the discount factor in the last three equations is to account for the latent components in the computation of x_j^k. Another alternative is to use the number of such late components expected to arrive during the present (kth) cycle. Denoting by U_j^k this (conditional) expectation as computed at the start of the kth cycle, we set

$$x_j^k = [h_j - s_j^k - U_j^k]_+, \tag{2.9}$$

and note that this control law does not involve the discount factor α_j.

With the first three approaches we have $2m$ parameters, $(H, A) := (h_1, \ldots, h_m, \alpha_1, \ldots, \alpha_m)$; in the fourth approach, we have only m variables, $H = (h_1, \ldots, h_m)$. Correspondingly, the optimization problem becomes $min\{g_K(H, A)\}$ or $min\{g_K(H)\}$, where $g_K = \mathbb{E}[G_K]$, and the dependence on the variables is via Eqns. (2.1)-(2.5), and either (2.6), (2.7), (2.8) or (2.9), respectively. In the forthcoming we will consider only (2.7) and (2.9).

3 Continuous-Parameter Modeling and Numerical Solution Methods

We clearly have here a stochastic optimization problem that does not yield itself to easy numerical solution techniques. Before describing our modeling and algorithmic approach to it, we summarize the development of the equations defining the cost optimization problem; Eq. (2.7) will be used for inventory control.

Dynamic equations:

$$s_j^k = s_j^{k-1} + z_j^{k-1} - \sum_{i=1}^{n} y_i^k \ell_{i,j}, \tag{3.1}$$

$$v_i^k = v_i^{k-1} + d_i^k - y_i^{k-1}. \tag{3.2}$$

Control equations:

$$y_1^k = min\{d_1^k + v_1^k, [\frac{s_j^{k-1} + z_j^{k-1}}{\ell_{1,j}}] : j = 1, \ldots, m\}, \tag{3.3}$$

and for all $i = 2, \ldots, n$,

$$y_i^k = min\{d_i^k + v_i^k, [\frac{s_j^{k-1} + z_j^{k-1} - \sum_{v=1}^{i-1} \ell_{v,j} y_v^k}{\ell_{i,j}}] : j = 1, \ldots, m\}; \tag{3.4}$$

$$x_j^k = [h_j - s_j^k - \alpha_j u_j^k]_+. \tag{3.5}$$

Output equation and cost-measure definition:

$$G_K = G(H, A) = K^{-1} \sum_{k=1}^{K} [\sum_{i=1}^{n} b_i v_i^k + \sum_{j=1}^{m} a_j s_j^k], \tag{3.6}$$

$$g_K = g_K(H, A) = \mathbb{E}[G_K]. \tag{3.7}$$

The variables are $(H, A) = (h_1, \ldots, h_m, \alpha_1, \ldots, \alpha_m)$, and z_j^k is determined by the probabilistic law of the components' yield times.

These equations do not lend themselves easily to a solution of the optimization problem

$$min\{g_K(H, A)\}$$

by numerical techniques. The main reason is that the function g_K may have many local minima and, in fact, exhibit considerable jitter as the variables H or A vary. This is mainly caused by the "integer part" in Eqns. (3.3)-(3.4). To see this point, consider the simple case of one part type and one component type and, for example, let $\ell_{1,1} = \ell = 5$. Suppose that, for a given fixed k, and for a specific sample path, $s_1^{k-1} + z_1^{k-1} = 7$. Suppose also that the backlogging cost dominates the inventory-holding cost, and correspondingly, $v_1^k >> s_1^{k-1} + z_1^{k-1}$, and hence, in Eq. (3.3),

$$y_1^k = [\frac{s_1^{k-1} + z_1^{k-1}}{\ell}] = [\frac{7}{5}] = 1.$$

Next suppose that to reduce the total cost we have to increase the production rate, that is, reduce the backlogging, and this can be done by increasing h_1 or reducing α_1 (see Eq. (3.5)). By changing either one of these parameters, let us say that $s_1^{k-1} + z_1^{k-1} = 8$. Now the inventory cost has been increased, but not enough components have been added in order to reduce the backlogging, since by Eq. (3.3),

$$y_1^k = [\frac{8}{5}] = 1.$$

The net result is an increase in the total cost.

To reduce the backlogging, we have to further increase the inventory until $s_1^{k-1} + z_1^{k-1} = 10$ since, in this case,

$$y_1^k = [\frac{10}{5}] = 2,$$

and more products have been assembled. Generally, the graph of G_K as a function of h_1 for a fixed α_1 can have the form shown in Fig. 3.1. We certainly can expect this jitter to be even more pronounced when multiple parts and products are involved.

To ameliorate this problem, we can get rid of the "integer parts" in the control equations, which now assume the following form.

Modified Control equations:

$$y_1^k = min\{d_1^k + v_1^k, \frac{s_j^{k-1} + z_j^{k-1}}{\ell_{1,j}} : j = 1, \ldots, m\}, \tag{3.8}$$

and for all $i = 2, \ldots, n$,

$$y_i^k = min\{d_i^k + v_i^k, \frac{s_j^{k-1} + z_j^{k-1} - \sum_{v=1}^{i-1} \ell_{v,j} y_v^k}{\ell_{i,j}} : j = 1, \ldots, m\}; \tag{3.9}$$

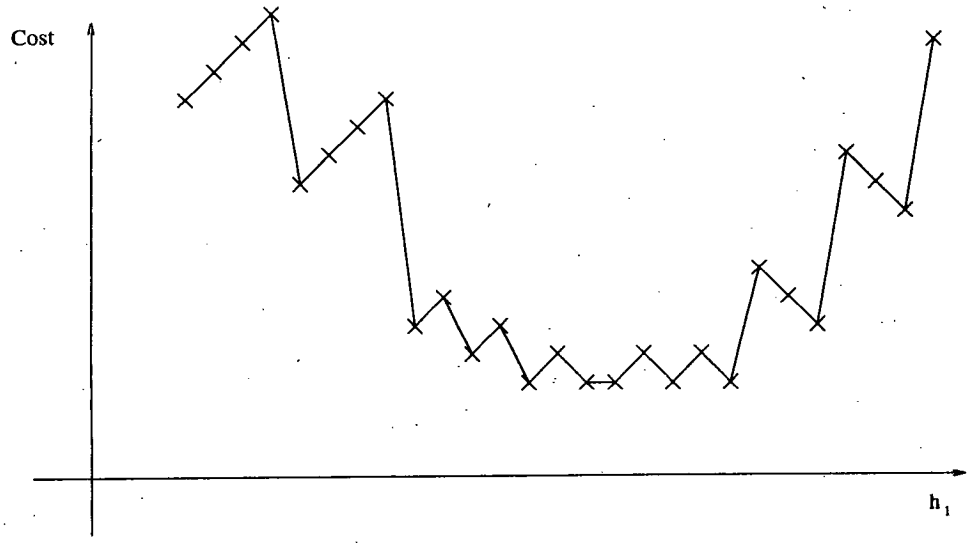

Fig. 3.1

$$x_j^k = (h_j - s_j^k - \alpha_j u_j^k)_+. \tag{3.10}$$

Consequently, the control variables $\{y_i^k\}$ and $\{x_j^k\}$, and hence the state variables $\{s_j^k\}$ and $\{v_i^k\}$, are allowed to have non-integer values. This approximation permits non-integer parts to be considered, and although some error with respect to the discrete model is bound to result, we later will show by numerical examples that it is generally quite small for our simulation experiments. At the same time, it is apparent that the continuous-parameter approximate model, and the resulting optimization problem, are much more amenable to numerical techniques.

With the modified control-equations (3.8)-(3.10), the only remaining integer variable is h_j, but there is no apparent reason not to allow it to have real values as well. The program

$$min\{g_K(H, A)\}$$

now become a continuous-variable, stochastic optimization problem for which gradient-descent methods can be applied. Once an algorithm stops, the final point can be slightly modified to take on the nearest integer values for the variables h_j.

To describe our algorithmic approach, we first present the problem in a more general context in order to simplify the notation used. Let $\ell_j(\theta) : \mathbb{R}^d \to \mathbb{R}$, $j = 1, 2, \ldots$, be a sequence of random functions defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and suppose that, w.p.1, as $j \to \infty$,

$$\ell_j(\theta) \to \ell(\theta) \tag{3.11}$$

pointwise, where $\ell(\theta)$ is a deterministic function. For example, let $L(\theta) = L(\theta; \omega)$ be a random function such that, for every θ, the strong law of large numbers applies. That is, for a fixed θ, w.p.1,

$$N^{-1} \sum_{n=1}^{N} L(\theta; \omega_n) \to \mathbb{E}[L(\theta)];$$

here $L(\theta; \omega_n)$, $n = 1, 2, \ldots$, denote mutually independent realizations of $L(\theta)$, and $\mathbb{E}$ denotes expectation. For a monotone-increasing sequence $\{N_j\}$, let $\ell_j(\theta) := N_j^{-1} \sum_{n=1}^{N_j} L(\theta; \omega_n)$, and let $\ell(\theta) := \mathbb{E}[L(\theta)]$.

This setup is quite common in simulation-based optimization where $\ell(\theta)$ does not have closed-form expressions, and it is approximated by averages of results obtained from independent Monte Carlo experiments. In our case, $\theta = (H, A)$, $L(\theta) = G_K(H, A)$, and $\ell(\theta) = g_K(H, A)$.

Consider the stochastic program

$$min\{\ell(\theta)\},$$

where $\ell(\theta)$ is approximated by $\ell_j(\theta)$ via (3.11). The computation of $\ell_j(\theta)$ depends on the sample path that is drawn, and suppose that $\nabla \ell_j(\theta)$ also can be computed from the same sample path. Infinitesimal Perturbation Analysis (IPA), for example, is a common technique for computing such sample gradients in discrete event dynamic systems, often with very low computing efforts; see, e.g., [7]. Let us suppose that, w.p.1, as $j \to \infty$,

$$\nabla \ell_j(\theta) \to \nabla \ell(\theta) \tag{3.12}$$

as well.

With the approximations (3.11)-(3.12) to the function and its derivative, gradient-descent algorithms can be applied. Such an algorithm typically would compute an iteration-sequence

$\{\theta_i\}$ that converges to a minimum (possibly local) of ℓ. A typical algorithm has the following form.

Algorithm 3.1. Given an iterate θ_i, compute the next iterate, θ_{i+1}, in the following way.

1. Determine an approximation-index $j = j_i$.
2. Draw a sample path, and compute the resulting sample gradient $\nabla \ell_j(\theta_i)$.
3. Compute a positive stepsize λ_i.
4. Set $\theta_{i+1} = \theta_i - \lambda_i \nabla \ell_j(\theta_i)$. □

Two questions of practical interest arise: (i) how to compute the precision index j_i in Step 1, and (ii) how to compute the stepsize λ_i in Step 3. Regarding the first question, we note that $j = j_i$ typically acts to balance the conflicting requirements of precision vs. short simulation runs, as larger values of j give better precision but often involve longer simulation (and hence computing) efforts.

The most prevalent algorithmic method is *stochastic approximation* (SA) [9], whereby j_i can be arbitrarily small, say a constant, and hence the algorithm spends little computing times at the iteration points. This necessitates, on the other hand, a-priori upper-bounds on the rate at which the stepsize-sequence $\{\lambda_i\}$ converge to 0, like the condition $\sum_{i=1}^{\infty} \lambda_i^2 < \infty$. Thus, unless such an algorithm approaches the desired minima at the early stage of its execution, its progress may become slow.[6]

An emerging alternative technique shifts the balance in favor of large stepsizes at the expense of larger computing efforts as compared to SA; see [13]. This method controls the precision index while performing an approximate line minimization against the direction of the computed sample gradient at each iteration point. The basic idea is this: at a given iteration θ_i, compute a precision index $j = j_i$ and draw a sample path. Compute the sample gradient $\nabla \ell_j(\theta_i)$, and then do an approximate line minimization in the direction $-\nabla \ell_j(\theta_i)$, starting from θ_i, to compute the next iterate θ_{i+1}. The entire procedure of computing θ_{i+1}

[6] Various modifications to improve the convergence speed certainly exist.

from θ_i is performed with the same sample path, i.e., the same seed in a Monte Carlo experiment. At the early stage of an algorithm's run, when the iterates are far from the optimal points, the algorithm would tend to take long steps between consecutive iteration points. At the same time, since the magnitude of the gradient can be quite large, the descent direction need not be exactly against the gradient $\nabla\ell(\theta_i)$, and consequently, the precision level and the associated precision index $j = j_i$ need not be very high. In other words, we start the algorithm with short simulations and typically obtain fast progress toward the minima. Naturally, as solution points are approached, the precision level must be increased. This algorithm was used for solving optimization problems in discrete event dynamic systems in [14] and, there as well as in this paper, rapid progress towards minima was obtained in few iterations with very short simulations. Its convergence properties have been rigorously established in [13].

The numerical procedure just described fits within the framework of Algorithm 3.1. At Step 3 we do the approximate line minimization. That is, we compute an approximation to

$$argmin\{\ell_j(\theta_i - \lambda\nabla\ell_j(\theta_i)) \ : \ \lambda \geq 0\},$$

and this approximate argmin will be λ_i. Recall that we use the same sample path throughout this procedure, namely, the entire line-function $\ell(\theta_i - \lambda\nabla\ell_j(\theta_i))$, $\lambda \geq 0$, as well as $\nabla\ell_j(\theta_i)$, are computed with the same seed. Thus, we in fact have here a *deterministic* function once the sample path is drawn, and therefore we can use some of the powerful nonlinear programming methods for computing the stepsize. The technique we have adopted for this purpose is called the *Armijo stepsize routine*, a detailed discussion of which can be found in [12], and a summary, later in the appendix.

We mention that a similar principle regarding simulation-based optimization has been developed in [11, 15], where nonlinear-programming techniques were applied in conjunction with precision control to the stochastic context of DEDS, and with very effective results.

Let us return to Algorithm 3.1. Regarding Step 1, we use some heuristic rules for improving the precision. Starting with fairly crude precision and hence short computing times, we keep the same seed for a number of iterations until observing that the descent obtained between two consecutive iteration points is small. It is then that we change the seed (and hence the sample path) and increase the precision level, etc.. In other words, the decision to

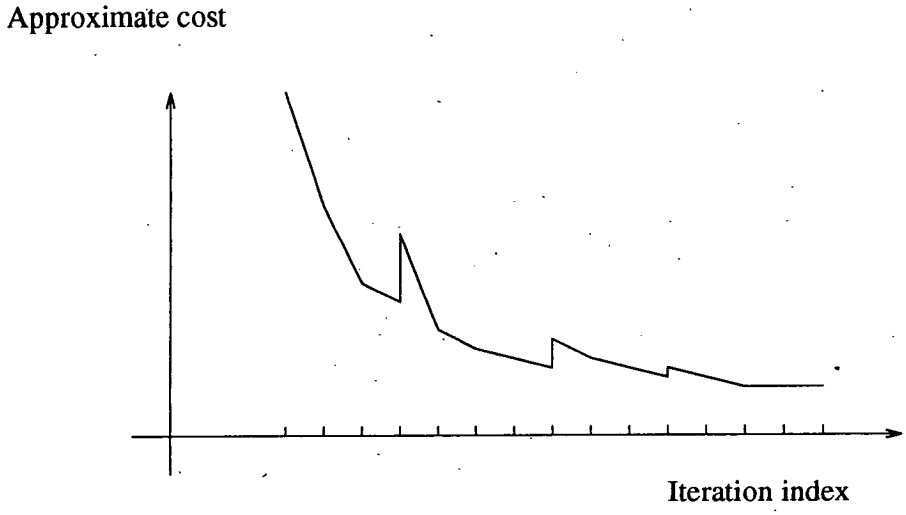

Fig. 3.2

increase the precision index j is based on the observed descent $\ell_j(\theta_{i+1}) - \ell_j(\theta_i)$, with $j = j_i$, according to heuristics regarding the question when such a descent is "small". While the precision index remains the same over a number of iterations, the same sample path is kept, and hence we have a deterministic function. Once the precision level is increased we recompute the value of the new approximating function with a new sample path, and continue from there. The graph of the approximate-function values vs. the number of iterations typically has the form shown in Fig. 3.2: between updates of the seed we obtain descents, and when precision is increased at an iterate, a slight jump up can be expected since the algorithm acted to minimize the approximating function used prior to the change. All of this requires an interactive computing environment in which, at each iteration, the program asks a few questions, including whether to increase the precision level, and if so, what should be the number of independent realizations over which the averages should be taken. Recall that our variable is $\theta = (H, A)$. Since H and A generally are not of the same scale, it may be advantageous to take steps against the partial gradients $\partial G_K/\partial H$ and $\partial G_K/\partial A$, separately. One of the questions asked by the program at each iteration is whether the step taken should be against the gradient, or either one of the partial gradients.

4 Numerical Experiments

The optimization algorithm was tried on a number of test problems of increasing complexity, using the continuous-variable approximate models described in Section 3. For each problem, two forms of inventory control were considered, one, according to Eq. (2.7), namely $x_j^k = [h_j - s_j^k - \alpha_j u_j^k]_+$, and the other, according to Eq. (2.9), i.e. $x_j^k = [h_j - s_j^k - U_j^k]_+$. Recall that, in Eq. (2.7) u_j^k denotes the number of type-j parts ordered before the start of the kth cycle and not yet arrived, whereas in (2.9), U_j^k is the latent type-j parts expected to arrive during the kth cycle. Observe that in the former case the variable is (H, A), and in the latter case, it consists of H alone.

[3] has addressed the optimization problem with Eq. (2.6) for inventory control, and hence the variable was (H, A). That reference used the discrete model directly, and applied a stochastic approximation algorithm to the discrete variables. It was noted there that optimization with respect to H while A was kept to a constant value ran much faster than optimization with respect to A with H being a constant. A similar behavior of the two variables was also noted in [10], where (2.7) was used for inventory control. Our numerical experiments, below, bring up the same point, and an explanation of this phenomenon will follow.

Since we minimize the performance functions derived from the approximate model based on continuous variables, a question of interest is to what extent the analogous functions, based on the discrete model, also would be minimized. To answer this question we provided, in the tables below, a run of the discrete-variable functions alongside the approximate-model function values, by using the same seeds, so that comparisons can be made. Indications are that, the minimum of the continuous-parameter approximate model, computed by the algorithm, is quite close to the minimum of the discrete model.

The computing environment, run on a SUN-SPARC station, includes an interactive software program permitting the user to change some control parameters between successive iterations. The program provides an output list (hence called the *output*) at the end of each iteration, based on which a decision concerning the control variables (*input*) can be made. The output list includes the iteration-point, i.e. (H, A) or H, as appropriate, the simulated

cost $G_K(H, A)$ of the continuous model, denoted by *Cost*; the square-magnitude of its partial gradients $\frac{\partial G_K}{\partial H}$ and $\frac{\partial G_K}{\partial A}$, denoted by $||\nabla_H||^2$ and $||\nabla_A||^2$, respectively (in the case where the inventory control is based on Eq. (2.9) only $\frac{\partial G_K}{\partial H}$ is relevant); the cost computed according to the discrete model (D_{cost}), and the stepsize with which the current iteration point was entered, λ. We have used the Armijo stepsize routine (please see the appendix) according to which $\lambda = 0.5^k$ for some integer k, and what the output provides is the value of k.

For each problem, the simulated system runs over a horizon of 60 cycles ($K = 60$). One such run corresponds to one realization, but to get the functions and gradients involved in the course of an iteration, we need to average the relevant quantities over a number of independent realizations; this set of realizations constitutes a single sample path, corresponding to j_i at Step 1 of the algorithm. Let us denote by N the number of such independent realizations, over whose averages the sample-performance functions and their gradients are computed (i.e., $N = j_i$ at Step 1). Clearly, larger values of N result in better precision and larger computing efforts than smaller values of N, as discussed in the previous section. For all of the problems we started with $N = 5$, and generally increased N based on some conclusions drawn from the output data, and especially when it was observed that successive reductions in the cost *Cost* were small and the stepsizes were small, i.e., k large. This is clearly evident from the tables below. The seeds were changed only when N was modified and hence, as discussed in the last section, the algorithm acts to minimize a deterministic function between successive updates of the precision-index N.

Each one of the problems below is described by the following data: n and m - the numbers of product- and component-types, respectively; an n-dimensional vector d - the product demands at each cycle (we use constant demands for simplicity's sake, random demands naturally can be incorporated without changing the algorithm); an n-dimensional vector b - backlogging costs; an m-dimensional vector a - inventory holding costs; an $n \times m$ matrix L - the assembly matrix; and a matrix with m rows, P, representing the probability distributions of the parts' yield times, that is, $P_{j,i}$ is the probability that type-j components ordered at the start of the k cycle arrive during the $k + i - 1$ cycle.

Problem 1. $n = 1$, $m = 3$, $d = 10$, $b = 10$, $a = (2, 1, 1)$, $L = (2\ 4\ 2)$, and

$$P = \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0.7 & 0.2 & 0.1 \\ 0.8 & 0.2 & 0.0 \end{pmatrix}$$

Table 4.1.1(a) shows a run of the algorithm, with the initial point $H = (100.0, 100.0, 100.0)$ and $A = (0.5, 0.5, 0.5)$. Whenever a new sample path was drawn, the algorithm computed the resulting costs and partial gradients' magnitudes, and the corresponding value of k in the table is *init*; this is the case for iteration numbers 1, 19, 28 and 36. At all other iterations, k is the value, according to the Armijo stepsize rule, at which the iteration point has been entered. In other words, the iterate (H, A) was computed from the previous iteration point with the stepsize of $1/2^k$ (please see the appendix). At each iteration, the algorithm moved against either one of the partial gradients. Starting with optimizing with respect to H while keeping A constant, it proceeded in this way through iteration number 14, thereafter it switched over to optimizing with respect to A while keeping H constant, etc.. The table shows that A is kept to a constant value between iterations 1 and 14, then H is a constant to iteration 18, and this indicates which variable, H or A, the algorithm optimizes with respect to. Whenever we change the seed, and increase N at the same time, we continue by optimizing with respect to H until the observed descent, namely the difference between the current value of *Cost* and the one at the previous iteration becomes small; it is then that we switch over to A. This could be done adaptively while the program was running because the lines of the table were shown on the screen one at a time during the course of the algorithm.

The following observations were made: (i) Most of the reduction in the cost (*Cost*), from 267.57 at iteration 1 to 96.09 at iteration 39, was obtained by the first 10 iterations, with $N = 5$, and most of the approach of H toward its final value also was made during these iterations. (ii) The relative difference between *Cost* and D_{cost} did not exceed 5%. (iii) $||\nabla_H||^2$ declines to 0 quite quickly for a given fixed sample path while $||\nabla_A||^2$ remains much larger, and in fact, optimization with respect to H typically runs much more smoothly than with respect to A. In particular, large values of $||\nabla_H||^2$ typically resulted in large descent in *Cost*, whereas large $||\nabla_A||^2$ sometimes resulted in very small descent.

To corroborate the results in Table 4.1.1(a), we ran the algorithm a second time with different seeds and a starting point (H, A). The results, presented in Table 4.1.1(b), show

even faster convergence while exhibiting similar characteristics to the earlier run, and the final points in the two respective runs, and their corresponding simulated costs, are quite close to each other.

We next ran the algorithm with Eq. (2.9) for inventory control, that is, $x_j^k = [h_j - s_j^k - U_j^k]_+$. Here, of course, the variable A is absent, and the free control parameter is H. The results are shown in Tables 4.1.2(a) and 4.1.2(b) for two independent runs with different starting points. Both runs appear to converge to final iteration points that are quite close to each other. For comparison's sake, we chose the same initial iteration points as in the algorithm's run for the previous case, where the variable was (H, A). That is, the initial H is the same in Tables 4.1.1(a) and 4.1.2(a), and in Tables 4.1.1(b) and 4.1.2(b), respectively. We observe that convergence is considerably faster for the case where (2.9) is involved as opposed to (2.7), namely the absence of A in the variable parameter results in a faster run of the algorithm. This suggests, once more, some adverse influence of the variable A, a point that will be later explained. Noted in particular is the fact that, in Table 4.1.2(b), the cost goes down from 2941.69 to under 100 in 13 iterations, and with $N = 5$. We also notice that the relative discrepancy between $Cost$ and D_{cost} goes up to over 10%. To check whether the solution obtained for the continuous model is close to a solution of the discrete model, we simulated the graphs of $Cost$ and D_{cost} in a small interval about the obtained minimum, as functions of h_1, h_2 and h_3, respectively. The simulations used the same seeds at all points, and were based on 50 samples each. The results of variations in h_1, while h_2 and h_3 were kept fixed, are shown in Fig. 4.1, indicating that, in spite of the discrepancy, the two minima are fairly close to each other. The graphs of variations in h_2 and h_3 yielded similar results, and are not shown here.

Problem 2. $n = 2$, $m = 4$, $d = (7, 12)$, $b = (13, 9)$, $a = (2, 1, 1, 2)$,

$$L = \begin{pmatrix} 2 & 4 & 2 & 3 \\ 5 & 1 & 4 & 2 \end{pmatrix},$$

and

$$P = \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0.7 & 0.2 & 0.1 \\ 0.8 & 0.2 & 0.0 \\ 0.1 & 0.6 & 0.3 \end{pmatrix}.$$

When Eq. (2.7) was used for the inventory control, numerical results for two independent runs with different starting points are shown in Tables 4.2.1(a) and 4.2.1(b), respectively. Similar observations as for the last problem can be made here as well: the final values of H and $Cost$ are quite close to each other, most of the approach toward the obtained optimum is made in few iterations (about 15) and with a small value of N (starting with $N = 5$, then increasing N to 20), the relative difference between $Cost$ and D_{cost} is upper-bounded by 5%, $||\nabla_H||^2$ goes to 0 rapidly while $||\nabla_A||^2$ remains large, and optimization with respect to A generally is too slow to be practical. Two independent runs for the case where the inventory control is determined by Eq. (2.9) are presented in Tables 4.2.2(a) and 4.2.2(b), and similar conclusions as for the last problem can be drawn: convergence is faster than for the case involving (2.7) where A is a part of the variable, but the relative discrepancy between $Cost$ and D_{cost} is about 10%. Finally, plots of $Cost$ and D_{cost} about the optimum with 200 realizations comprising of the sample path, with respect to variations in h_1, are shown in Fig. 4.2, and suggest that the two respective minima are quite close to each other. Similar plots with respect to the other components of H were also made, and exhibited similar results, but are not shown here.

Our last problem is larger than the first two and it appears to exhibit nondifferentiabilities and possible non-uniqueness of the local minima.

Problem 3. $n = 3$, $m = 7$, $d = (12, 9, 7)$, $b = (13, 12, 8)$, $a = (4, 2, 6, 1, 4, 5, 2)$,

$$L = \begin{pmatrix} 4 & 7 & 0 & 8 & 2 & 1 & 1 \\ 0 & 0 & 3 & 3 & 2 & 1 & 6 \\ 2 & 4 & 1 & 6 & 0 & 2 & 1 \end{pmatrix},$$

and

$$P = \begin{pmatrix} 1.0 & 0 & 0 \\ 0.5 & 0.5 & 0 \\ 0.7 & 0.2 & 0.1 \\ 0.6 & 0.6 & 0.2 \\ 0.4 & 0.3 & 0.3 \\ 0.1 & 0.7 & 0.2 \\ 0.4 & 0.4 & 0.2 \end{pmatrix}.$$

For the case using (2.7) for inventory control, two independent runs are shown in Tables 4.3.1(a) and 4.3.1(b), respectively. The two obtained solutions are fairly close to each other but by no means identical, and it is impossible to say whether they represent multiple local minima. It is noted, though, that most of the cost reduction is done at the early phases of the algorithm's runs, when the number of statistical realizations is relatively small. For the second case, where inventory control is computed by (2.9) and the variable is H, two results are shown in Tables 4.3.2(a) and 4.3.2(b). Here we observe a faster computation, and closer proximity of the two obtained solutions than in the first case, moreover, all of the observations we made for the first two problems can be seen here as well.

The numerical results suggest two possible conclusions: (i) the algorithm's strength lies in its rapid progress during the initial phase of its run toward the solution. Indeed, most of the descent in the cost was obtained early, and with very few independent realizations per sample path. Once a solution point is approached, greater precision, and hence increasing simulation run times, are required. (ii) Optimization with respect to H runs much more smoothly than with respect to A. In fact, the variable A appears to be problematic: $||\nabla_A||$ fails to decline to 0 even in iterations close to the optimum, and generally, stepsizes taken in the direction $-\frac{\partial G_K}{\partial A}$ are very small and the resulting reduction in *Cost* is insignificant. As earlier noted, a similar phenomenon was noticed in [3]; we now suggest an explanation.

In view of Eqns. (3.1)-(3.6), G_K involves the operations of *min* and *max*, and hence the sample-performance function is nondifferentiable in the variable (H, A). It is piecewise differentiable though, and has a gradient at each point lying in the interior of one of the differentiability regions. In crossing over from one such region to another, the signs of the partial derivatives may change. Let us examine two partial derivatives, with respect to h_j and α_j, for a fixed j. Observe Eq. (3.5), the continuous-variable analogue of (2.7), and suppose that $x_j^k = h_j - s_j^k - \alpha_j u_j^k > 0$. The partial derivatives $\frac{\partial G_K}{\partial h_j}$ and $\frac{\partial G_K}{\partial \alpha_j}$ involve $\frac{\partial x_j^k}{\partial h_j}$ and $\frac{\partial x_j^k}{\partial \alpha_j}$, respectively, and by (3.5), $\frac{\partial x_j^k}{\partial h_j} = 1$, and $\frac{\partial x_j^k}{\partial \alpha_j} = u_j^k$. Now $|u_j^k|$ typically is of the same order-of-magnitude as h_j, about 100, and therefore, $||\nabla_A||$ is two orders-of-magnitude larger than $||\nabla_H||$. Consequently, a small variation in the parameter A can throw an iteration-point (H, A) off into a different differentiability region, which is a common cause of the failure of gradient-descent algorithms. Moreover, this high sensitivity of the performance function with respect to A may have adverse effects on the system's performance at a design point,

leading us to suggest that the inventory-control equation be like (2.9), namely not using the parameter A.

5 Conclusions

This paper has presented a technique for solving supply management problems in cyclic assembly systems that manufacture multiple product types from a common components' bank. Parts' yield times are assumed to be uncertain, causing the buildup of inventories and products' backlogging. The main part of the problem is how to formulate a component delivery schedule so as to minimize the sum of inventory costs and backlogging costs.

In a raw form, the resulting optimization problem includes the production schedule and the quantities of components ordered at each cycle, and thus it has many discrete-parameter variables. Consequently, we have adopted an approach consisting of two steps: first, the determination of the production schedules according to a fixed policy, and second, a formulation of a set of control variables that greatly reduces the dimensionality of the associated optimization problem. This results in a parametrized set of inventory control policies, from which the one giving the lowest cost is sought. The modified optimization problem still can be quite difficult, due to the discrete nature of the dynamic variables. We have approached it by remodeling of the system by continuous-parameter dynamics, and applying to the resulting optimization problem a powerful stochastic gradient-descent algorithm.

In summary, what originally had been formulated as a fairly intractable optimal design problem has been addressed by careful modeling in such a way that effective optimization techniques could be easily coded and executed. Numerical results testified to the viability of the whole approach in that they exhibited rapid movement of the algorithm toward the optimal solutions.

6 Appendix

This section briefly presents the steepest descent algorithm with Armijo stepsizes and discusses some aspects of its implementation, for a comprehensive discussion, please see [12]. That algorithm has been extensively applied to deterministic optimization problems, and more recently, adapted to the stochastic environment of discrete event dynamic systems (e.g., [14]). To set the stage, let $\ell : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function which one wishes to minimize, to the extent of finding its stationary points. The algorithm computes an iteration-sequence $\{\theta_i\}_{i=1}^{\infty}$ provably convergent to stationary points [12] by performing approximate line-minimization against the gradient directions. Suppose it is at an iteration-point θ_i; it computes the next point, θ_{i+1}, in the following way.

Given two constants $\alpha \in (0,1)$ and $\beta \in (0,1)$.

1. Compute $d_i := \nabla\ell(\theta_i)$, if $d_i = 0$ then stop, a stationary point has been found.
2. Compute an integer k_i.
3. Compute the integer $k(i)$ defined by

$$k(i) \;=\; min\{k \geq k_i \;:\; \ell(\theta_i - \beta^k d_i) - \ell(\theta_i) \leq -\alpha\beta^k ||d_i||^2\}. \tag{6.1}$$

4. Set $\theta_{i+1} = \theta_i - \beta^{k(i)} d_i$. □

Some explanation is due. Step 1 is self evident, and Step 2 will be explained shortly. Consider Step 3. The objective is to take a "good" step from θ_i in the direction $-d_i = -\nabla\ell(\theta_i)$ that yields a good descent in the value of ℓ. Let us denote the stepsize by λ_i, so that the next iteration point is $\theta_{i+1} = \theta_i - \lambda_i d_i$.

Let us fix a sequence $\{\lambda(k)\}_{k=1}^{\infty}$ monotonically decreasing to 0, and suppose that λ_i must be an element from this sequence. Fig. 6.1 shows how λ_i is computed: in it we see two graphs of functions of λ, $\lambda > 0$. The linear graph is of the function $\xi_1(\lambda) := \ell(\theta_i) - \alpha\lambda||d_i||^2$, and the other graph is of the function $\xi_2 := \ell(\theta_i - \lambda d_i)$. Observe that $\xi_1'(0) = -\alpha||d_i||^2$ whereas $\xi_2'(0) = -||d_i||^2$, and hence, since $\alpha \in (0,1)$, we have that $\xi_1'(0) > \xi_2'(0)$. Consequently, for small-enough but positive λ, $\xi_1(\lambda) > \xi_2(\lambda)$. For larger λ the latter inequality may be

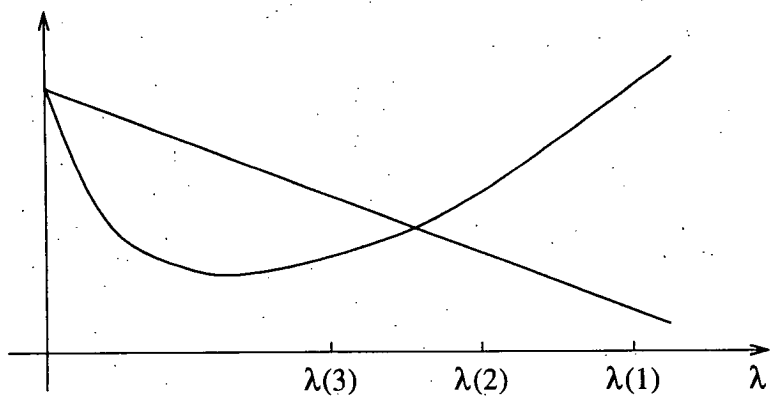

Fig. 6.1

reversed, as is shown in the figure, or it might not. The Armijo stepsize, λ_i, is the largest $\lambda(k)$, $k = 1, 2, \ldots$, for which $\xi_1(\lambda(k)) \geq \xi_2(\lambda(k))$, and by the above discussion, λ_i is well defined and positive. In the figure, it can be seen that $\lambda_i = \lambda(3)$.

Based on an extensive experience with this algorithm [12] it is customary to set $\lambda_k = \beta^k$, for some $\beta \in (0, 1)$, which explains the term in Eq. (6.1) at Step 3. Commonly used values of α and β are $\alpha = \beta = 0.5$, as we have done in our numerical experiments. The search for λ_i from the set $1/2^k$ typically starts at at some $k = k_i$, and proceeds by increasing k. A common rule-of-thumb [12] is to set $k_i = k(i-1) - 1$ or $k_i = k(i-1) - 2$, the latter alternative being taken in Section 4. Finally, we mention one of the appealing properties of this algorithm: large gradients $\nabla\ell(\theta_i)$ and stepsizes typically result in large descents in ℓ between consecutive iterations. The reason can be seen in Steps 3 and 4, from which it follows that $\ell(\theta_{i+1}) - \ell(\theta_i) \leq -\alpha\lambda_i ||d_i||^2$. This was shown, in Section 4, for optimization with respect to H, but not with regard to A, as earlier explained.

References

[1] M. Caramanis and G. Liberopoulos, "Perturbation Analysis for the Design of Flexible Manufacturing System Flow Controllers", *Operations Research*, Vol. 40, pp. 1107-1125,

1992.

[2] C. Chu, J.M. Proth, and X.L. Xie, "Supply Management in Assembly Systems: The Basic Problem", Research Report #1624, INRIA, 1992.

[3] C. Chu, J.M. Proth, Y. Wardi, and X.L. Xie, "Supply Management in Assembly Systems: The Case of Random Yield Times", presented at the 11^{th} Intl. Conf. on Analysis and Optimization of Systems, Antibes, France, June 15-17, 1994.

[4] M. Cohen and H. Lee, "Resource Development Analysis of Global Manufacturing and Distribution Networks", *Journal of Manufacturing and Operations Management*, Vol. 2, pp. 81-104, 1989.

[5] C.K. Hahn, K.H. Kim, and J.S. Kim, "Costs of Competition: Implications for Purchasing Strategy", *Journal of Purchasing and Materials Management*, Vol. 22, pp. 2-7, 1986.

[6] T.E. Hendrick and W.A. Rush, "Determining Performance Appraisal Criteria for the Buyer", *Journal of Purchasing and Materials Management*, Vol. 24, pp. 18-26, 1988.

[7] Y.C. Ho and X.R. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems*, Kluwer Academic Publishers, Boston, MA, 1991.

[8] J. Kimemia and S. Gershwin, "An Algorithm for the Computer Control of Production in Flexible Manufacturing Systems", *IIE Transactions*, Vol. 15, pp. 353-362, 1983.

[9] H.J. Kushner and D.S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, Springer-Verlag, New York, NY, 1978.

[10] G. Mauroy and Y. Wardi, "Cost Optimization in Supply Management Policies for Assembly Systems with Random Component Yield Times", in *Proc. Conf. on Emerging Technologies and Factory Automation*, Paris, France, October 10-13, 1995, to appear.

[11] E. L. Plambeck, B.R. Fu, S.M. Robinson, and R. Suri, "Optimizing Performance Functions in Stochastic Systems", to appear.

[12] E. Polak, *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, NY, 1971.

[13] A. Shapiro and Y. Wardi, "Convergence Analysis of Stochastic Algorithms", *Mathematics of Operations Research*, to appear.

[14] Y. Wardi and K. Lee, "Applications of Descent Algorithms with Armijo Stepsizes to Simulation-based Optimization of Queueing Networks", in *Proc. 30th Conf. on Decision and Control*, pp. 110-115, Brighton, England, December 1991.

[15] D. Yan and H. Mukai, "An Optimization Algorithm with Probabilistic Simulation", *Journal of Optimization Theory and Applications*, to appear.

i	H	A	N	Cost	D_{cost}	$\| \nabla_H \|^2$	$\| \nabla_A \|^2$	k
1	(100.00, 100.00, 100.00)	(0.50, 0.50, 0.50)	5	267.57	267.38	5.46	1224.03	init
2	(98.07, 99.10, 99.03)	(0.50, 0.50, 0.50)	5	262.14	262.42	5.39	1177.07	0
3	(90.36, 95.66, 95.17)	(0.50, 0.50, 0.50)	5	240.58	240.20	5.39	1157.51	-2
4	(59.53, 81.87, 79.70)	(0.50, 0.50, 0.50)	5	155.03	156.26	4.61	890.29	0
5	(52.60, 78.59, 75.83)	(0.50, 0.50, 0.50)	5	137.67	140.49	3.64	740.32	-2
6	(39.75, 75.75, 68.10)	(0.50, 0.50, 0.50)	5	114.06	114.40	1.01	157.52	1
7	(40.29, 75.83, 66.17)	(0.50, 0.50, 0.50)	5	112.24	112.54	1.12	152.55	-1
8	(38.56, 75.83, 62.30)	(0.50, 0.50, 0.50)	5	109.26	111.26	2.08	68.28	-1
9	(40.63, 76.40, 60.37)	(0.50, 0.50, 0.50)	5	106.74	107.94	1.18	139.88	0
10	(38.83, 76.46, 56.42)	(0.50, 0.50, 0.50)	5	103.07	105.26	0.99	141.80	-2
11	(43.84, 77.34, 41.36)	(0.50, 0.50, 0.50)	5	93.17	94.74	1.65	307.89	0
12	(41.62, 76.72, 40.22)	(0.50, 0.50, 0.50)	5	90.46	92.12	0.51	103.98	-1
13	(40.22, 76.92, 40.05)	(0.50, 0.50, 0.50)	5	89.61	91.33	0.36	92.82	0
14	(40.09, 76.96, 40.09)	(0.50, 0.50, 0.50)	5	89.52	91.33	0.10	155.02	2
15	(40.09, 76.96, 40.09)	(0.58, 0.55, 0.47)	5	88.54	90.83	0.06	69.74	7
16	(40.09, 76.96, 40.09)	(0.58, 0.55, 0.47)	5	88.54	90.83	0.06	69.75	5
17	(40.09, 76.96, 40.09)	(0.60, 0.57, 0.46)	5	88.30	89.98	0.07	49.48	7
18	(40.09, 76.96, 40.09)	(0.60, 0.57, 0.45)	5	88.27	89.96	0.07	49.76	11
19	(40.09, 76.96, 40.09)	(0.60, 0.57, 0.45)	20	99.78	103.50	9.56	2435.90	init
20	(41.50, 77.56, 40.31)	(0.60, 0.57, 0.45)	20	96.52	103.06	1.91	493.02	1
21	(42.67, 78.27, 40.51)	(0.60, 0.57, 0.45)	20	95.34	101.52	0.22	60.27	0
22	(43.09, 79.95, 39.79)	(0.60, 0.57, 0.45)	20	94.71	96.80	0.24	67.09	-2
23	(43.04, 80.00, 40.02)	(0.60, 0.57, 0.45)	20	94.61	96.80	0.26	64.27	5
24	(43.03, 79.99, 40.01)	(0.60, 0.57, 0.45)	20	94.61	96.80	0.26	64.26	4
25	(43.03, 79.99, 40.01)	(0.58, 0.62, 0.48)	20	94.29	95.92	0.20	9.50	7
26	(43.03, 79.99, 40.01)	(0.58, 0.62, 0.48)	20	94.29	95.92	0.20	9.50	5
27	(43.03, 80.00, 40.00)	(0.58, 0.62, 0.48)	20	94.29	95.92	0.08	9.50	6
28	(43.03, 80.00, 40.00)	(0.58, 0.62, 0.48)	50	99.34	98.59	0.33	116.37	init
29	(43.59, 80.12, 40.00)	(0.58, 0.62, 0.48)	50	99.04	97.30	0.23	74.37	0
30	(44.03, 80.01, 40.13)	(0.58, 0.62, 0.48)	50	98.86	97.05	0.08	19.27	0
31	(44.09, 79.98, 40.11)	(0.58, 0.62, 0.48)	50	98.85	97.32	0.19	19.11	2
32	(44.09, 79.98, 40.11)	(0.49, 0.58, 0.58)	50	98.47	97.15	0.11	28.41	5
33	(44.09, 79.98, 40.11)	(0.52, 0.58, 0.60)	50	98.32	96.90	0.11	6.79	6
34	(44.09, 79.98, 40.11)	(0.52, 0.58, 0.61)	50	98.30	96.96	0.11	6.11	4
35	(44.09, 79.98, 40.11)	(0.52, 0.61, 0.76)	50	98.04	96.73	0.11	2.72	4
36	(44.09, 79.98, 40.11)	(0.52, 0.61, 0.76)	100	96.23	98.89	0.14	21.20	init
37	(44.07, 80.04, 40.04)	(0.52, 0.61, 0.76)	100	96.21	98.89	0.10	23.41	2
38	(44.05, 80.02, 40.01)	(0.52, 0.61, 0.76)	100	96.20	98.90	0.11	23.85	3
39	(44.05, 80.02, 40.01)	(0.53, 0.58, 0.77)	100	96.09	98.62	0.13	3.32	7

Table 4.1.1(a): Problem 1, inventory controlled by Eq. (2.7), first run

i	H	A	N	Cost	D_{cost}	$\| \nabla_H \|^2$	$\| \nabla_A \|^2$	k
1	(100.00, 3.00, 7.00)	(0.50, 0.50, 0.50)	5	2901.89	3108.42	4559.01	22041.80	init
2	(99.39, 19.87, 6.76)	(0.50, 0.50, 0.50)	5	2273.02	2331.99	5980.24	6734.89	2
3	(99.13, 21.54, 16.27)	(0.50, 0.50, 0.50)	5	1850.99	1916.53	2496.50	128979.00	3
4	(98.14, 46.14, 20.50)	(0.50, 0.50, 0.50)	5	895.51	963.61	4379.15	101320.00	1
5	(97.65, 50.72, 36.39)	(0.50, 0.50, 0.50)	5	289.97	381.97	533.11	114506.00	2
6	(97.17, 56.07, 38.50)	(0.50, 0.50, 0.50)	5	214.43	219.85	35.08	10565.70	2
7	(95.23, 61.14, 40.87)	(0.50, 0.50, 0.50)	5	193.11	195.90	5.14	1870.62	0
8	(87.48, 65.69, 39.68)	(0.50, 0.50, 0.50)	5	173.94	175.93	4.26	1397.42	-2
9	(56.60, 76.73, 43.44)	(0.50, 0.50, 0.50)	5	118.11	119.48	2.89	392.16	0
10	(50.75, 75.79, 40.09)	(0.50, 0.50, 0.50)	5	107.82	108.64	1.43	243.33	-2
11	(46.25, 76.06, 38.53)	(0.50, 0.50, 0.50)	5	103.72	105.16	0.80	284.96	2
12	(45.47, 76.25, 38.92)	(0.50, 0.50, 0.50)	5	102.92	105.47	0.78	283.49	0
13	(42.40, 76.90, 40.53)	(0.50, 0.50, 0.50)	5	100.76	103.73	0.24	141.74	-2
14	(40.49, 80.14, 39.55)	(0.50, 0.50, 0.50)	5	99.54	99.53	1.59	118.76	1
15	(40.49, 80.14, 39.55)	(0.54, 0.57, 0.53)	5	98.84	99.08	2.06	58.93	7
16	(40.49, 80.14, 39.55)	(0.54, 0.57, 0.53)	5	98.84	99.08	2.06	58.93	5
17	(40.49, 80.14, 39.55)	(0.54, 0.57, 0.53)	20	98.21	100.03	3.46	336.77	init
18	(41.09, 80.36, 40.24)	(0.54, 0.57, 0.53)	20	97.16	100.54	0.17	53.51	1
19	(41.89, 80.54, 40.11)	(0.54, 0.57, 0.53)	20	96.96	98.86	0.03	104.12	-1
20	(42.24, 80.03, 39.99)	(0.54, 0.57, 0.53)	20	96.90	98.11	0.58	117.89	-1
21	(42.24, 80.03, 39.99)	(0.58, 0.56, 0.54)	20	96.53	97.67	0.50	26.69	8
22	(42.24, 80.03, 39.99)	(0.58, 0.56, 0.54)	20	96.53	97.67	0.50	26.69	15
23	(42.24, 80.03, 39.99)	(0.58, 0.56, 0.54)	50	98.24	101.34	0.76	116.44	init
24	(42.25, 80.03, 40.01)	(0.58, 0.56, 0.54)	50	98.23	101.34	0.11	116.47	5
25	(42.27, 80.03, 40.02)	(0.58, 0.56, 0.54)	50	98.22	101.34	0.11	116.53	3
26	(42.27, 80.03, 40.02)	(0.56, 0.52, 0.54)	50	97.87	100.88	0.03	14.05	8
27	(42.27, 80.03, 40.02)	(0.56, 0.52, 0.54)	100	96.41	97.04	0.17	78.27	init

Table 4.1.1(b): Problem 1, inventory controlled by Eq. (2.7), second run

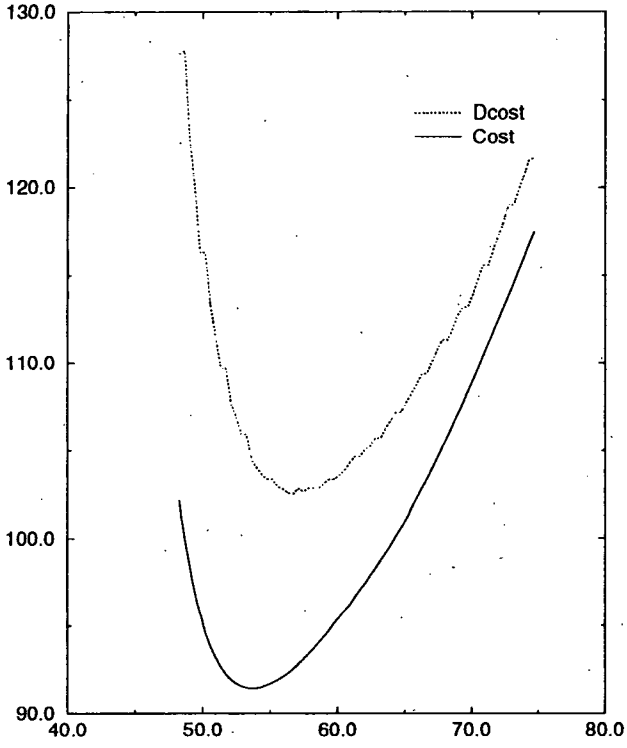
130.0
120.0
110.0
100.0
90.0
40.0
50.0
60.0
70.0
80.0
Dcost
Cost

Fig. 4.1

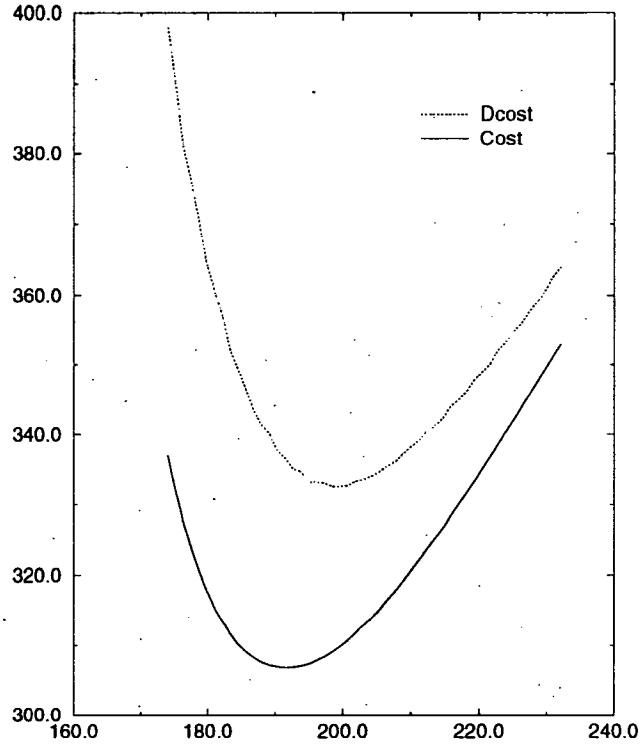
400.0
380.0
360.0
340.0
320.0
300.0
160.0
180.0
200.0
220.0
240.0
Dcost
Cost

Fig. 4.2

i	H	N	Cost	D_{cost}	$\| \nabla_H \|^2$	k
1	(100.00, 100.00, 100.00)	5	209.12	209.86	5.59	init
2	(98.10, 101.02, 99.04)	5	203.50	204.60	5.54	0
3	(90.50, 105.03, 95.18)	5	183.20	182.81	4.54	-2
4	(60.10, 105.93, 79.77)	5	116.04	118.85	1.30	0
5	(59.13, 108.17, 75.92)	5	111.13	113.95	1.26	-2
6	(50.04, 109.31, 60.50)	5	100.32	109.42	14.03	0
7	(51.71, 110.15, 60.36)	5	96.50	103.04	3.85	1
8	(52.56, 110.64, 60.29)	5	95.41	101.74	2.81	1
9	(53.22, 111.13, 60.18)	5	94.66	100.66	1.87	1
10	(53.34, 111.25, 60.15)	5	94.54	99.78	1.31	3
11	(53.53, 111.46, 60.10)	5	94.37	99.82	1.07	2
12	(53.53, 111.46, 60.10)	20	94.29	102.86	0.52	init
13	(53.43, 111.18, 59.44)	20	93.76	103.02	0.33	0
14	(53.43, 110.32, 57.30)	20	92.83	104.33	0.28	-2
15	(53.43, 110.32, 57.30)	20	92.83	104.33	0.28	15
16	(53.43, 110.32, 57.30)	50	90.96	105.27	0.07	init
17	(53.52, 110.08, 57.26)	50	90.89	105.33	0.06	0
18	(53.83, 109.14, 57.12)	50	90.66	105.24	0.04	-2

Table 4.1.2(a): Problem 1, inventory controlled by Eq. (2.9), first run

i	H	N	Cost	D_{cost}	$\| \nabla_H \|^2$	k
1	(100.00, 3.00, 7.00)	5	2941.69	3049.85	3315.14	init
2	(99.75, 10.19, 6.88)	5	2681.93	2759.67	2099.27	3
3	(99.26, 21.29, 9.67)	5	2395.04	2452.67	4987.54	2
4	(99.20, 21.72, 11.84)	5	2298.71	2352.04	1994.46	5
5	(99.08, 23.83, 13.66)	5	2215.64	2301.51	1889.23	3
6	(98.11, 43.11, 23.65)	5	1561.59	1641.90	1905.27	1
7	(96.18, 77.67, 50.24)	5	341.07	425.06	1021.90	0
8	(95.86, 85.30, 52.59)	5	197.47	243.16	265.69	2
9	(95.67, 87.22, 53.24)	5	178.44	197.37	69.16	3
10	(95.27, 89.14, 53.94)	5	169.21	176.88	25.04	2
11	(94.37, 91.17, 55.09)	5	162.59	167.35	7.18	1
12	(90.71, 94.89, 56.33)	5	151.87	151.28	4.28	-1
13	(60.32, 107.79, 53.88)	5	95.96	99.48	2.13	-3
14	(54.95, 106.08, 55.42)	5	88.72	98.42	0.58	9
15	(54.95, 106.08, 55.42)	5	88.72	98.42	0.58	15
16	(54.95, 106.08, 55.42)	20	89.60	103.34	0.83	init
17	(54.92, 106.58, 56.18)	20	89.04	101.60	0.20	0
18	(54.57, 107.24, 56.66)	20	88.84	100.41	0.11	-1
19	(54.20, 107.68, 56.32)	20	88.65	101.04	0.04	0
20	(54.20, 107.68, 56.32)	20	88.65	101.04	0.04	15
21	(54.20, 107.68, 56.32)	50	89.33	102.79	0.03	init
22	(54.03, 107.65, 56.30)	50	89.26	102.81	0.01	0
23	(53.62, 107.63, 56.33)	50	89.15	103.31	0.00	-2
24	(53.62, 107.63, 56.33)	50	89.15	103.31	0.00	12

Table 4.1.2(b): Problem 1, inventory controlled by Eq. (2.9), second run

i	H	A	N	Cost	D_{cost}	$\| \nabla_H \|^2$	$\| \nabla_A \|^2$	k
1	(200.00, 200.00, 200.00, 200.00)	(0.50, 0.50, 0.50, 0.50)	5	724.39	723.80	9.02	24575.00	init
2	(198.13, 199.03, 199.03, 198.08)	(0.50, 0.50, 0.50, 0.50)	5	715.37	715.69	9.02	24550.50	0
3	(190.67, 195.17, 195.17, 190.41)	(0.50, 0.50, 0.50, 0.50)	5	679.33	679.18	8.87	22778.20	-2
4	(132.27, 164.29, 164.23, 129.06)	(0.50, 0.50, 0.50, 0.50)	5	417.28	418.40	4.74	15365.40	-1
5	(126.44, 156.57, 157.28, 116.36)	(0.50, 0.50, 0.50, 0.50)	5	388.68	394.75	2.75	8898.64	-2
6	(134.42, 148.85, 151.01, 112.72)	(0.50, 0.50, 0.50, 0.50)	5	375.47	378.39	1.72	4452.13	1
7	(133.66, 146.92, 149.50, 112.18)	(0.50, 0.50, 0.50, 0.50)	5	372.16	375.56	1.55	3728.87	-1
8	(129.38, 131.48, 137.96, 109.64)	(0.50, 0.50, 0.50, 0.50)	5	352.57	357.76	1.60	3327.25	-3
9	(131.86, 131.42, 137.60, 109.94)	(0.50, 0.50, 0.50, 0.50)	5	350.94	353.60	0.23	1256.21	10
10	(131.86, 131.42, 137.60, 109.94)	(0.51, 0.52, 0.51, 0.53)	5	350.13	353.16	0.14	538.99	10
11	(131.86, 131.42, 137.60, 109.94)	(0.51, 0.52, 0.51, 0.53)	5	350.13	353.16	0.14	538.99	15
12	(131.86, 131.42, 137.60, 109.94)	(0.51, 0.52, 0.51, 0.53)	20	361.67	369.33	11.17	15188.40	init
13	(134.39, 130.48, 137.51, 111.92)	(0.51, 0.52, 0.51, 0.53)	20	351.79	358.95	6.75	9292.02	0
14	(136.39, 129.71, 137.45, 113.38)	(0.51, 0.52, 0.51, 0.53)	20	347.22	352.81	1.64	1219.38	0
15	(139.21, 126.52, 135.53, 115.50)	(0.51, 0.52, 0.51, 0.53)	20	342.54	346.37	1.03	2695.43	-2
16	(140.36, 120.32, 130.54, 114.32)	(0.51, 0.52, 0.51, 0.53)	20	338.37	345.21	0.42	4233.27	1
17	(141.38, 120.89, 130.41, 114.86)	(0.51, 0.52, 0.51, 0.53)	20	337.71	342.88	0.19	5222.07	-1
18	(142.59, 122.02, 129.90, 114.83)	(0.51, 0.52, 0.51, 0.53)	20	337.03	340.03	0.10	5358.64	-1
19	(142.59, 122.02, 129.90, 114.83)	(0.53, 0.47, 0.51, 0.58)	20	334.27	336.83	0.79	620.75	10
20	(142.59, 122.02, 129.90, 114.83)	(0.53, 0.47, 0.51, 0.58)	20	334.27	336.83	0.79	620.75	15
21	(142.59, 122.02, 129.90, 114.83)	(0.53, 0.47, 0.51, 0.58)	50	335.01	337.48	0.91	783.96	init
22	(143.17, 121.90, 129.88, 115.57)	(0.53, 0.47, 0.51, 0.58)	50	334.56	337.24	0.22	801.60	0
23	(144.10, 121.07, 128.58, 116.07)	(0.53, 0.47, 0.51, 0.58)	50	333.88	338.17	0.09	1524.20	-2
24	(146.64, 116.67, 120.30, 115.78)	(0.53, 0.47, 0.51, 0.58)	50	330.79	333.19	0.06	1102.35	-1
25	(146.64, 116.65, 120.30, 115.80)	(0.53, 0.47, 0.51, 0.58)	50	330.78	333.18	0.20	2639.62	4
26	(146.64, 116.65, 120.30, 115.80)	(0.53, 0.47, 0.51, 0.58)	100	342.59	346.04	1.45	728.23	init
27	(147.35, 116.59, 121.00, 116.48)	(0.53, 0.47, 0.51, 0.58)	100	341.39	346.41	0.45	794.47	0
28	(148.55, 116.18, 122.84, 117.96)	(0.53, 0.47, 0.51, 0.58)	100	340.19	345.39	0.14	2217.16	-2
29	(149.87, 114.60, 125.00, 117.55)	(0.53, 0.47, 0.51, 0.58)	100	339.46	342.82	0.07	4153.61	1
30	(149.87, 114.60, 125.00, 117.55)	(0.57, 0.48, 0.52, 0.62)	100	336.55	340.33	0.34	382.35	10
31	(149.87, 114.60, 125.00, 117.55)	(0.57, 0.48, 0.52, 0.62)	100	336.55	340.33	0.34	382.35	15
32	(150.10, 114.49, 125.15, 118.06)	(0.57, 0.48, 0.52, 0.62)	100	336.26	340.30	0.13	802.22	0
33	(150.48, 114.22, 125.20, 118.61)	(0.57, 0.48, 0.52, 0.62)	100	336.06	339.88	0.06	1353.67	-1
34	(150.48, 114.22, 125.20, 118.61)	(0.57, 0.48, 0.52, 0.62)	200	337.29	340.74	0.12	1239.73	init

Table 4.2.1(a): Problem 2, inventory controlled by Eq. (2.7), first run

i	H	A	N	Cost	D_{cost}	$\| \nabla_H \|^2$	$\| \nabla_A \|^2$	k
1	(20.00, 200.00, 10.00, 15.00)	(0.50, 0.50, 0.50, 0.50)	5	4872.14	4914.36	2998.24	221141.00	init
2	(20.29, 199.51, 25.20, 37.76)	(0.50, 0.50, 0.50, 0.50)	5	3791.96	3850.41	2721.81	345418.00	1
3	(43.07, 199.03, 31.73, 48.66)	(0.50, 0.50, 0.50, 0.50)	5	3053.09	3056.91	411.01	133687.00	1
4	(61.57, 197.09, 56.63, 74.69)	(0.50, 0.50, 0.50, 0.50)	5	2277.35	2311.78	414.42	354336.00	-1
5	(189.58, 189.36, 123.10, 149.92)	(0.50, 0.50, 0.50, 0.50)	5	513.00	514.71	7.70	24370.90	-2
6	(162.37, 173.89, 115.93, 119.25)	(0.50, 0.50, 0.50, 0.50)	5	401.10	402.09	3.59	16138.70	0
7	(157.85, 170.02, 114.84, 114.69)	(0.50, 0.50, 0.50, 0.50)	5	389.04	392.97	1.84	13098.60	-2
8	(152.55, 162.29, 115.37, 109.22)	(0.50, 0.50, 0.50, 0.50)	5	381.58	388.09	3.41	475.65	1
9	(153.95, 160.36, 117.52, 111.05)	(0.50, 0.50, 0.50, 0.50)	5	376.34	381.06	1.27	3033.24	-1
10	(156.06, 144.89, 126.01, 107.88)	(0.50, 0.50, 0.50, 0.50)	5	363.07	367.28	0.56	7488.97	-3
11	(155.34, 146.19, 126.11, 107.78)	(0.50, 0.50, 0.50, 0.50)	5	362.43	366.35	1.04	8579.00	10
12	(155.34, 146.19, 126.11, 107.78)	(0.50, 0.50, 0.50, 0.50)	20	363.24	365.12	2.21	8934.88	init
13	(154.37, 145.30, 125.59, 107.33)	(0.50, 0.50, 0.50, 0.50)	20	361.06	363.26	1.93	9092.60	0
14	(150.86, 141.76, 123.59, 105.87)	(0.50, 0.50, 0.50, 0.50)	20	354.09	357.57	1.27	8127.72	-2
15	(139.91, 127.44, 124.11, 105.83)	(0.50, 0.50, 0.50, 0.50)	20	341.10	344.10	2.54	320.77	0
16	(141.40, 125.75, 124.78, 107.98)	(0.50, 0.50, 0.50, 0.50)	20	337.71	343.47	0.74	2382.45	-1
17	(142.62, 119.06, 125.19, 108.79)	(0.50, 0.50, 0.50, 0.50)	20	333.55	336.82	0.11	6090.55	-2
18	(143.12, 117.83, 125.13, 108.90)	(0.50, 0.50, 0.50, 0.50)	20	333.24	336.05	0.01	6935.18	2
19	(143.12, 117.83, 125.13, 108.90)	(0.50, 0.50, 0.50, 0.50)	50	340.76	345.17	1.51	1337.68	init
20	(144.01, 117.97, 125.28, 109.73)	(0.50, 0.50, 0.50, 0.50)	50	339.35	344.59	1.30	1314.36	0
21	(144.73, 118.11, 125.47, 110.57)	(0.50, 0.50, 0.50, 0.50)	50	338.51	342.26	0.25	1704.10	0
22	(145.33, 117.84, 125.26, 111.30)	(0.50, 0.50, 0.50, 0.50)	50	338.19	341.41	0.12	2471.64	-1
23	(145.42, 114.62, 122.37, 114.60)	(0.50, 0.50, 0.50, 0.50)	50	337.26	339.96	0.13	4709.80	-3
24	(145.31, 114.64, 122.41, 114.74)	(0.50, 0.50, 0.50, 0.50)	50	337.23	339.96	0.05	5136.67	12
25	(145.31, 114.64, 122.41, 114.74)	(0.50, 0.50, 0.50, 0.50)	100	342.90	346.70	0.22	6599.40	init
26	(145.54, 114.61, 122.64, 114.41)	(0.50, 0.50, 0.50, 0.50)	100	342.69	348.15	0.14	5799.49	0
27	(146.37, 114.38, 123.37, 113.41)	(0.50, 0.50, 0.50, 0.50)	100	342.28	348.36	0.03	2849.31	-2
28	(146.37, 114.38, 123.37, 113.41)	(0.50, 0.50, 0.50, 0.50)	200	339.76	344.96	0.08	5277.66	init
29	(146.47, 114.21, 123.58, 113.42)	(0.50, 0.50, 0.50, 0.50)	200	339.68	342.83	0.07	5124.99	0
30	(146.89, 113.60, 124.29, 113.51)	(0.50, 0.50, 0.50, 0.50)	200	339.52	341.53	0.10	5277.15	-2

Table 4.2.1(b): Problem 2, inventory controlled by Eq. (2.7), second run

i	H	N	Cost	D_{cost}	$\| \nabla_H \|^2$	k
1	(200.00, 200.00, 200.00, 200.00)	5	510.79	506.24	8.10	init
2	(198.36, 199.05, 199.05, 198.10)	5	502.74	499.00	7.84	0
3	(192.14, 195.23, 195.23, 190.52)	5	471.58	468.42	7.28	-2
4	(170.30, 179.96, 179.96, 160.17)	5	375.27	385.91	6.28	0
5	(176.33, 176.15, 178.11, 153.38)	5	354.96	361.06	3.95	-2
6	(177.86, 160.88, 168.19, 127.35)	5	322.71	348.06	26.97	0
7	(179.77, 160.40, 168.72, 128.96)	5	314.20	335.15	12.13	1
8	(180.90, 159.92, 168.91, 130.18)	5	310.93	326.46	4.09	1
9	(181.69, 159.45, 169.15, 130.52)	5	309.86	325.88	2.50	1
10	(182.91, 158.49, 169.45, 130.48)	5	308.34	323.43	1.21	0
11	(184.54, 154.68, 170.04, 129.15)	5	304.04	319.57	1.03	-2
12	(185.02, 147.04, 171.20, 126.66)	5	299.70	315.63	2.21	1
13	(185.72, 146.75, 171.25, 127.94)	5	298.27	313.67	0.56	0
14	(186.01, 146.42, 171.43, 128.52)	5	297.99	311.06	0.10	0
15	(186.01, 146.42, 171.43, 128.52)	20	313.49	346.76	1.01	init
16	(186.35, 145.60, 171.77, 128.84)	20	312.83	344.83	0.74	0
17	(186.87, 142.31, 172.52, 128.37)	20	310.95	342.71	0.12	-2
18	(188.33, 137.98, 175.44, 127.33)	20	309.68	338.94	0.20	0
19	(188.33, 137.98, 175.44, 127.33)	20	309.68	338.94	0.20	11
20	(188.33, 137.98, 175.44, 127.33)	50	305.94	333.30	0.23	init
21	(188.47, 137.65, 175.19, 127.12)	50	305.68	333.42	0.20	0
22	(189.10, 136.33, 174.35, 126.47)	50	304.86	334.46	0.12	-2
23	(195.41, 127.91, 171.06, 125.96)	50	302.67	332.44	0.03	-1
24	(193.38, 128.24, 172.58, 126.95)	50	302.41	332.52	0.02	-3
25	(193.38, 128.24, 172.58, 126.95)	50	302.41	332.52	0.02	15
26	(193.38, 128.24, 172.58, 126.95)	100	306.98	336.05	0.22	init
27	(193.38, 128.24, 172.58, 126.95)	100	306.98	336.10	0.22	10
28	(193.38, 128.24, 172.58, 126.95)	200	305.79	335.91	0.18	init
29	(193.36, 128.13, 172.74, 127.32)	200	305.69	335.11	0.08	0

Table 4.2.2(a): Problem 2, inventory controlled by Eq. (2.9), first run

i	H	N	Cost	D_{cost}	$\| \nabla_H \|^2$	k
1	(20.00, 200.00, 10.00, 15.00)	5	5072.25	5118.13	3002.48	init
2	(19.04, 199.52, 29.98, 33.71)	5	4321.10	4400.56	4748.63	1
3	(27.50, 199.40, 30.32, 35.28)	5	3944.47	4044.79	1728.82	3
4	(40.12, 198.91, 36.04, 50.77)	5	3280.75	3358.57	546.80	1
5	(64.34, 196.98, 64.38, 78.94)	5	2608.08	2681.13	373.36	-1
6	(178.08, 189.25, 166.90, 98.64)	5	655.12	771.03	503.48	-2
7	(185.21, 188.28, 168.58, 119.82)	5	360.65	440.18	17.01	4
8	(185.87, 188.04, 168.75, 120.56)	5	356.56	434.35	21.60	2
9	(189.15, 187.07, 169.42, 123.64)	5	345.26	408.69	2.27	0
10	(191.38, 183.20, 168.68, 127.62)	5	340.03	389.70	1.22	-2
11	(191.17, 175.47, 167.52, 123.5)	5	334.03	403.11	2.35	1
12	(191.41, 173.54, 168.32, 125.73)	5	330.64	395.99	0.60	-1
13	(192.52, 168.44, 168.36, 129.10)	5	327.83	391.52	0.94	-2
14	(192.37, 165.81, 167.84, 126.31)	5	325.60	397.15	0.37	2
15	(192.51, 165.81, 167.89, 126.35)	5	325.55	397.19	0.43	2
16	(192.51, 165.81, 167.89, 126.35)	20	327.91	355.05	1.53	init
17	(192.54, 164.86, 168.22, 127.07)	20	326.74	352.97	1.72	0
18	(192.55, 162.96, 168.85, 128.75)	20	324.64	350.22	1.05	-1
19	(191.05, 147.69, 172.01, 133.57)	20	313.68	339.91	1.28	-3
20	(188.99, 145.29, 171.57, 130.37)	20	309.63	344.63	0.60	9
21	(188.99, 145.29, 171.57, 130.37)	20	309.63	344.63	0.60	15
22	(188.99, 145.29, 171.57, 130.37)	50	312.89	337.72	0.25	init
23	(189.21, 144.89, 171.66, 130.19)	50	312.61	338.15	0.25	0
24	(190.41, 143.52, 171.75, 129.33)	50	311.74	337.65	0.12	-2
25	(190.41, 143.52, 171.75, 129.33)	50	311.74	337.65	0.12	15
26	(190.41, 143.52, 171.75, 129.33)	100	315.58	344.56	0.27	init
27	(190.62, 143.18, 172.02, 129.14)	100	315.34	344.14	0.26	0
28	(191.52, 141.84, 173.14, 128.55)	100	314.58	341.79	0.17	-2
29	(192.05, 140.48, 173.86, 128.35)	100	314.10	340.54	0.12	2
30	(192.05, 140.48, 173.86, 128.35)	200	309.59	334.43	0.24	init
31	(191.87, 140.14, 173.68, 128.12)	200	309.30	335.01	0.17	0

Table 4.2.2(b): Problem 2, inventory controlled by Eq. (2.9), second run

i	H	A
1	(200.00, 200.00, 200.00, 200.00, 200.00, 200.00, 200.00)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
2	(196.13, 201.51, 194.14, 227.40, 196.33, 195.10, 204.60)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
3	(188.40, 214.00, 182.43, 273.27, 185.19, 185.33, 212.34)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
10	(118.56, 245.20, 77.50, 330.91, 113.74, 96.73, 204.52)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.5,0 0.50)
11	(118.56, 245.20, 77.50, 330.91, 113.74, 96.73, 204.52)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
12	(114.95, 244.11, 75.28, 331.85, 111.82, 95.45, 203.20)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
13	(114.11, 243.70, 75.79, 332.17, 111.87, 94.31, 203.50)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
14	(114.01, 243.66, 75.69, 332.20, 111.82, 94.28, 203.49)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
15	(114.01, 243.66, 75.69, 332.20, 111.82, 94.28, 203.49)	(0.50, 0.51, 0.51, 0.49, 0.51, 0.52, 0.51)
16	(114.02, 243.65, 75.71, 332.20, 111.83, 94.28, 203.51)	(0.50, 0.51, 0.51, 0.49, 0.51, 0.52, 0.51)
17	(114.02, 243.65, 75.71, 332.20, 111.83, 94.28, 203.51)	(0.50, 0.51, 0.51, 0.49, 0.51, 0.52, 0.51)
18	(110.28, 242.47, 73.27, 332.89, 109.02, 91.86, 202.58)	(0.50, 0.51, 0.51, 0.49, 0.51, 0.52, 0.51)
19	(110.28, 242.47, 73.26, 332.89, 109.01, 91.86, 202.58)	(0.50, 0.51, 0.51, 0.49, 0.51, 0.52, 0.51)
20	(110.28, 242.47, 73.26, 332.89, 109.01, 91.86, 202.58)	(0.50, 0.51, 0.51, 0.49, 0.51, 0.52, 0.51)
21	(109.74, 242.27, 73.15, 333.06, 109.10, 91.40, 202.60)	(0.50, 0.51, 0.51, 0.49, 0.51, 0.52, 0.51)
22	(109.74, 242.27, 73.15, 333.06, 109.10, 91.40, 202.60)	(0.50, 0.51, 0.51, 0.49, 0.51, 0.52, 0.51)

i	N	Cost	D_{cost}	$\| \nabla_H \|^2$	$\| \nabla_A \|^2$	k
1	5	4704.86	4711.50	860.77	3.3255e+06	init
2	5	4013.10	3993.36	684.34	2.56521e+06	0
3	5	3133.28	3126.19	174.31	368624.00	-1
10	5	1500.31	1510.32	58.35	257416.00	15
11	20	1487.93	1485.88	27.17	47439.40	init
12	20	1466.23	1456.42	41.91	45298.90	0
13	20	1456.30	1447.79	24.54	56387.60	2
14	20	1455.55	1446.45	24.02	62795.40	5
15	20	1451.30	1449.76	3133.98	4.20249e+07	13
16	20	1450.47	1445.09	26.39	134209.00	11
17	50	1415.51	1410.30	36.43	63558.80	init
18	50	1384.02	1376.24	3200.66	2.82286e+06	0
19	50	1383.61	1376.05	42.29	90613.90	12
20	100	1407.46	1397.00	37.92	55287.90	init
21	100	1405.02	1398.89	16.10	51669.30	3
22	200	1411.38	1395.93	45.94	76535.40	init

Table 4.3.1(a): Problem 3, inventory controlled by Eq. (2.7), first run

i	H	A
1	(10.00, 20.00, 30.00, 40.00, 50.00, 60.00, 70.00)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
2	(2.83, 20.55, 44.13, 68.88, 47.88, 55.11, 75.72)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
3	(13.34, 20.06, 46.46, 70.20, 49.33, 53.89, 79.88)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
4	(13.44, 28.35, 51.96, 81.99, 52.86, 49.07, 95.79)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
5	(18.81, 28.88, 53.15, 84.08, 54.20, 47.87, 99.33)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
6	(20.44, 33.61, 56.02, 98.58, 59.71, 43.08, 111.31)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
7	(21.54, 70.93, 67.34, 137.03, 93.44, 42.61, 137.59)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
8	(32.67, 72.39, 67.26, 137.74, 92.54, 43.08, 137.96)	(0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50)
26	(49.06, 93.41, 70.62, 165.46, 90.88, 50.94, 138.95)	(0.50, 0.11, 0.58, 0.20, 0.72, 0.58, 0.37)
27	(49.11, 94.16, 70.62, 165.49, 90.88, 50.94, 138.95)	(0.50, 0.11, 0.58, 0.20, 0.72, 0.58, 0.37)
28	(49.11, 94.16, 70.62, 165.49, 90.88, 50.94, 138.95)	(0.50, 0.11, 0.58, 0.20, 0.72, 0.58, 0.37)
29	(68.38, 97.14, 62.05, 173.10, 99.14, 59.91, 141.79)	(0.50, 0.11, 0.58, 0.20, 0.72, 0.58, 0.37)
43	(89.16, 152.06, 61.79, 229.56, 114.24, 75.27, 165.32)	(0.50, 0.09, 0.56, 0.08, 0.67, 0.55, 0.31)
44	(89.16, 152.06, 61.79, 229.56, 114.24, 75.27, 165.32)	(0.50, 0.09, 0.56, 0.08, 0.67, 0.55, 0.31)
45	(88.89, 151.79, 62.40, 229.82, 114.17, 75.50, 165.32)	(0.50, 0.09, 0.56, 0.08, 0.67, 0.55, 0.31)
51	(92.74, 154.61, 67.78, 239.66, 116.32, 81.50, 167.14)	(0.50, 0.08, 0.57, 0.08, 0.68, 0.55, 0.31)
52	(92.74, 154.61, 67.78, 239.66, 116.32, 81.50, 167.14)	(0.50, 0.08, 0.57, 0.08, 0.68, 0.55, 0.31)
53	(92.74, 154.61, 67.78, 239.66, 116.32, 81.50, 167.14)	(0.50, 0.09, 0.56, 0.04, 0.66, 0.57, 0.32)
67	(110.73, 195.73, 78.55, 282.30, 122.20, 109.12, 184.11)	(0.50, 0.09, 0.56, 0.04, 0.66, 0.57, 0.32)
68	(110.73, 195.73, 78.55, 282.30, 122.20, 109.12, 184.11)	(0.50, 0.09, 0.56, 0.04, 0.66, 0.57, 0.32)
69	(106.42, 195.77, 77.02, 286.09, 121.97, 107.78, 184.10)	(0.50, 0.09, 0.56, 0.04, 0.66, 0.57, 0.32)
70	(106.40, 195.77, 77.01, 286.12, 121.97, 107.78, 184.10)	(0.50, 0.09, 0.56, 0.04, 0.66, 0.57, 0.32)
71	(106.41, 195.77, 77.01, 286.12, 121.97, 107.78, 184.10)	(0.50, 0.09, 0.56, 0.04, 0.66, 0.57, 0.32)

Table 4.3.1(b): Problem 3, inventory controlled by Eq. (2.7), second run (1^{st} part)

i	N	Cost	D_{cost}	$\| \nabla_H \|^2$	$\| \nabla_A \|^2$	k
1	5	7605.95	7595.82	1146.72	240219.00	init
2	5	7003.83	7069.92	2219.64	229114.00	0
3	5	6689.07	6754.03	526.80	279433.00	2
4	5	6176.55	6254.54	810.93	205068.00	0
5	5	6067.86	6139.98	440.34	230836.00	2
6	5	5664.71	5711.77	301.80	213981.00	0
7	5	5050.38	5046.27	2041.12	124173.00	-2
8	5	4685.71	4721.77	1127.52	1.20468e+06	6
26	5	3854.98	3947.96	2436.71	237718.00	5
27	5	3845.40	3930.29	5739.43	375104.00	3
28	10	3798.83	3843.72	668.25	297852.00	init
29	10	3448.26	3474.88	465.38	487634.00	0
43	10	2171.71	2162.29	178.79	647776.00	6
44	20	2340.13	2307.00	655.03	287336.00	init
45	20	2328.73	2322.65	342.98	305910.00	5
51	20	2151.38	2154.20	581.16	296389.00	6
52	40	2190.45	2181.71	154.42	167208.00	init
53	40	2177.56	2183.28	372.14	327178.00	13
67	40	1552.47	1544.84	34.69	112256.00	2
68	100	1567.59	1556.75	37.10	56359.60	init
69	100	1547.58	1536.19	73.12	79568.70	0
70	100	1547.43	1536.03	99.46	117983.00	8
71	100	1547.38	1536.03	23.88	97802.00	10

Table 4.3.1(b): Problem 3, inventory controlled by Eq. (2.7), second run (2^{nd} part)

i	H	N	Cost	D_{cost}	$\| \nabla_H \|^2$	k
1	(200.00, 200.00, 200.00, 200.00, 200.00, 200.00, 200.00)	5	7157.86	7047.30	1662.32	init
2	(196.13, 191.88, 194.20, 238.18, 196.14, 195.23, 207.22)	5	5623.18	5625.79	1814.11	0
3	(192.27, 185.16, 188.40, 278.32, 192.27, 190.48, 215.71)	5	4683.08	4745.34	522.39	0
4	(176.80, 255.73, 165.20, 322.92, 176.85, 171.50, 212.14)	5	3359.81	3420.00	242.25	-2
5	(145.87, 296.45, 118.80, 409.34, 150.07, 133.55, 245.47)	5	1564.32	1651.45	626.59	1
6	(146.57, 296.60, 118.65, 409.32, 150.02, 133.40, 245.68)	5	1554.26	1645.00	127.11	5
7	(146.71, 296.92, 118.40, 409.80, 149.93, 133.12, 245.67)	5	1547.72	1620.08	130.99	3
8	(147.78, 299.50, 116.22, 413.60, 149.10, 130.88, 245.50)	5	1502.34	1560.43	350.20	1
9	(148.04, 299.46, 116.21, 413.62, 149.09, 130.82, 245.60)	5	1499.46	1556.18	339.85	6
10	(148.06, 299.46, 116.21, 413.62, 149.08, 130.81, 245.61)	5	1499.29	1556.18	339.85	6
11	(148.06, 299.46, 116.21, 413.62, 149.08, 130.81, 245.61)	20	1494.69	1572.50	82.66	init
12	(146.89, 301.96, 111.81, 417.61, 145.45, 125.69, 246.22)	20	1424.31	1502.93	43.91	0
13	(140.97, 302.08, 100.68, 432.39, 136.78, 109.93, 244.98)	20	1267.39	1382.23	13.51	-2
14	(142.98, 303.41, 103.00, 436.78, 135.29, 105.32, 245.29)	20	1249.23	1343.28	16.63	3
15	(141.65, 303.67, 102.91, 437.75, 134.80, 104.27, 245.17)	20	1243.00	1340.96	21.41	1
16	(138.27, 304.08, 101.56, 442.64, 128.57, 101.19, 244.94)	20	1218.87	1322.97	51.74	-1
17	(138.87, 304.00, 102.07, 442.96, 128.68, 101.45, 244.88)	20	1215.40	1326.68	130.71	4
18	(138.87, 304.00, 102.07, 442.96, 128.68, 101.45, 244.88)	20	1215.40	1326.68	130.71	15
19	(138.87, 304.00, 102.07, 442.96, 128.68, 101.45, 244.88)	50	1175.11	1279.15	24.74	init
20	(138.24, 303.77, 100.67, 443.66, 127.04, 102.18, 244.64)	50	1168.28	1275.73	12.15	1
21	(138.94, 303.38, 98.85, 445.27, 124.88, 102.83, 243.94)	50	1160.22	1273.56	8.31	0
22	(138.94, 303.38, 98.85, 445.27, 124.88, 102.83, 243.94)	50	1160.22	1273.56	8.31	15
23	(138.94, 303.38, 98.85, 445.27, 124.88, 102.83, 243.94)	100	1166.10	1260.79	6.26	init
24	(139.11, 303.44, 98.15, 445.80, 124.62, 102.42, 243.22)	100	1164.03	1252.82	86.06	1

Table 4.3.2(a): Problem 3, inventory controlled by Eq. (2.9), first run

i	H	N	Cost	D_{cost}	$\| \nabla_H \|^2$	k
1	(10.00, 20.00, 30.00, 40.00, 50.00, 60.00, 70.00)	5	8164.79	8194.84	3081.44	init
2	(6.08, 19.32, 30.90, 45.55, 49.52, 59.41, 70.35)	5	7958.88	8027.86	1062.23	3
3	(8.09, 18.37, 42.14, 56.37, 47.59, 57.03, 73.13)	5	7607.42	7681.70	729.97	1
13	(30.49, 57.45, 80.72, 135.99, 75.55, 46.15, 153.01)	5	5470.87	5539.07	1803.80	3
14	(30.49, 57.45, 80.72, 135.99, 75.55, 46.15, 153.01)	5	5470.87	5539.07	1803.80	15
15	(30.49, 57.45, 80.72, 135.99, 75.55, 46.15, 153.01)	10	5388.37	5479.99	376.03	init
16	(37.92, 61.28, 79.28, 139.87, 76.75, 45.08, 155.13)	10	5284.70	5379.23	297.22	1
35	(149.32, 258.60, 88.57, 368.89, 120.47, 92.34, 219.00)	10	1689.90	1824.31	866.03	4
36	(149.32, 258.60, 88.57, 368.89, 120.47, 92.34, 219.00)	10	1689.90	1824.31	866.03	15
37	(149.32, 258.60, 88.57, 368.89, 120.47, 92.34, 219.00)	20	1817.37	1932.49	402.32	init
38	(149.36, 258.68, 88.84, 369.00, 120.39, 92.34, 219.04)	20	1813.61	1931.77	375.71	6
41	(149.48, 259.00, 89.65, 369.47, 120.92, 92.39, 219.13)	20	1802.87	1914.00	487.90	15
42	(149.48, 259.00, 89.65, 369.47, 120.92, 92.39, 219.13)	40	1814.75	1973.77	2727.08	init
43	(149.48, 259.00, 89.65, 369.47, 120.92, 92.39, 219.13)	40	1814.75	1973.77	2727.08	15
44	(149.48, 259.00, 89.65, 369.47, 120.92, 92.39, 219.13)	40	1814.75	1973.77	2727.08	15
45	(149.48, 259.00, 89.65, 369.47, 120.92, 92.39, 219.13)	40	1814.75	1973.77	2727.08	15
46	(149.48, 259.00, 89.65, 369.47, 120.92, 92.39, 219.13)	50	1782.34	1961.75	312.74	init
47	(145.19, 261.47, 93.33, 375.51, 119.55, 93.70, 218.94)	50	1701.13	1899.73	203.77	1
56	(145.66, 294.12, 96.27, 445.17, 124.97, 99.37, 242.19)	60	1169.26	1257.85	61.21	7
57	(145.66, 294.12, 96.27, 445.17, 124.97, 99.37, 242.19)	60	1169.26	1257.85	61.21	5
58	(145.68, 294.15, 96.32, 445.19, 124.96, 99.37, 242.20)	60	1169.01	1257.85	27.91	6
59	(145.68, 294.15, 96.32, 445.19, 124.96, 99.37, 242.20)	100	1173.41	1277.11	109.33	init

Table 4.3.2(b): Problem 3, inventory controlled by Eq. (2.9), second run

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)

Unité de recherche INRIA Rennes - IRISA, Campus universitaire de Beaulieu 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes - 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 (France)
Unité de recherche INRIA Rocquencourt - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis - 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

ISSN 0249 - 6399